Andrei Piccinini Legg

PROPOSTA DE CÓDIGOS LDPC PARA CANAIS DE RESPOSTA PARCIAL

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

PROPOSTA DE CÓDIGOS LDPC PARA CANAIS DE RESPOSTA PARCIAL

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

Andrei Piccinini Legg

Florianópolis, março de 2007.

PROPOSTA DE CÓDIGOS LDPC PARA CANAIS DE RESPOSTA PARCIAL

Andrei Piccinini Legg

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em *Comunicações e Processamento de Sinais*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

_____
Bartolomeu Ferreira Uchôa Filho, Ph.D.

_____
Nelson Sadowski, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____
Bartolomeu Ferreira Uchôa Filho, Ph.D.
Presidente

_____
Carlos Aurélio Faria da Rocha, Dr.

_____
Leonardo Silva Resende, Dr.

_____
Joceli Mayer, Ph.D.

Aos meus Pais, a minha avó e ao meu tio. . .

AGRADECIMENTOS

Ao meu orientador, Ph. D. Bartolomeu Ferreira Uchôa Filho, pelo incentivo, pela inesgotável dedicação e sobretudo pela sua amizade.

A todos os outros membros do Gpqcom (professores e alunos) pelo apoio e companheirismo.

A todos os meus amigos que sempre me incentivaram durante estes anos de estudo.

Aos meus pais, minha avó e meu tio, por todo amor e apoio que me deram.

A todos,

Muito Obrigado

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

PROPOSTA DE CÓDIGOS LDPC PARA CANAIS DE RESPOSTA PARCIAL

Andrei Piccinini Legg

Março / 2007

Orientador: Bartolomeu Ferreira Uchôa Filho, Ph.D.
Área de Concentração: Comunicações e Processamento de Sinais.
Palavras-chave: Códigos corretores de erros, Códigos LDPC, Algoritmo de soma-produto (SPA), Canais de resposta parcial, Grafos Bipartidos.
Número de Páginas: 58.

RESUMO: Neste trabalho é apresentada uma proposta de códigos de verificação de paridade de baixa densidade (LDPC, *Low Density Parity-Check Codes*) para canais de resposta parcial. É proposta a utilização de um codificador de códigos RA (Repeat-Accumulate Codes) modificado com o objetivo de obter-se um casamento entre os canais de resposta parcial e as palavras-código submetidas a eles. A decodificação do código é realizada utilizando o algoritmo soma-produto (SPA). São também apresentados resultados de simulação que demonstram que para uma relação sinal-ruído (SNR) de 5dB é possível atingir uma probabilidade de erros de bit (BER) inferior a 10^{-5}, quando na ausência do código esse desempenho é conseguido com uma SNR de aproximadamente 10dB, evidenciando o excelente desempenho da proposta. Este trabalho encontra aplicação em, por exemplo, codificação para canais de gravação magnética.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

PROPOSITION OF LDPC CODES FOR PARCIAL RESPONSE CHANNELS

Andrei Piccinini Legg

March / 2007

Advisor: Bartolomeu Ferreira Uchôa Filho, Ph.D.
Area of Concentration: Communications and Signal Processing.
Keywords: Error correcting codes, LDPC codes, Sum-product Algorithm (SPA), Partial response Channels, Bipartite graphs.
Number of Pages: 58.

ABSTRACT: This work proposes Low Density Parity-Check Codes for Partial Response Channels. It proposes the use of an RA (Repeat-Accumulate) encoder, modified with the purpose of obtaining a match between the partial response channels and the codewords submitted to them. The decoding of the code is obtained through the use of the sum-product algorithm (SPA). Also presented are simulation results that demonstrate that for a signal-to-noise ratio (SNR) of 5dB it is possible to attain a bit error probability (BER) under 10^{-5}, while in the absence of the code the same result demands an SNR of 10dB, becoming evident the excellent performance of the proposal. This work finds application in, for instance, coding for magnetic recording channels.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

$\mathbf{J}$, $\mathbf{D}$	Grafos
$\mathbf{V}$	Conjunto de nós de um grafo
$\mathbf{A}$	Conjunto de arestas de um grafo
$\mathbf{Z}$	Caminho em um grafo
$\mathbf{C}$	Conjunto de palavras-código
$\mathbf{H}$	Matriz de verificação de paridade
$\mathbf{G}$	Matriz geradora de código de bloco
$\mathbf{a}$, $\mathbf{a}_1$, $\mathbf{a}_2$	Arestas de um grafo
$\mathbf{u}$	Vetor de informação
$\hat{\mathbf{u}}$	Vetor de informação estimado
$\mathbf{v}$	Vetor de palavra-código
$\hat{\mathbf{v}}$	Vetor de palavra-código estimado
$\mathbf{e}$	Vetor de erro
$\mathbf{r}$	Vetor recebido
$\mathbf{s}$	Vetor de síndrome
R	Taxa do código
E	Evento erro de palavra-código
n	Comprimento da palavra-código
k	Comprimento do bloco de informação
$x_0,\ldots,x_{11}$ $x'_0,\ldots,x'_{11}$ $f_1,\ldots,f_5$ $c_1,\ldots,c_9$ $q,q_1,\ldots,q_n$ $w,w_1,\ldots,w_n$	Nós de um grafo
p	Ordem de um grafo
m	Tamanho de um grafo
w_r	Peso de Hamming por linha da matriz

w_c	Peso de Hamming por coluna da matriz
b_k	Seqüência de bits de comprimento k
$x(t)$	Pulso retangular
$s(t)$	Função degrau unitário
$h(t)$	Pulso de Lorentzian
$i(t)$	Corrente elétrica
$r(t)$	Tensão elétrica na cabeça de leitura
$d_{\min}(\mathbf{C})$	Distância mínima de Hamming para um conjunto de palavras-código $\mathbf{C}$
$d_E^2(\mathbf{a},\mathbf{b})$	Distância euclidiana entre dois vetores **a** e **b**
$d_H(\mathbf{a},\mathbf{b})$	Distância de Hamming entre dois vetores **a** e **b**
$\omega_H(\mathbf{a})$	Peso de Hamming de um vetor **a**
$d^+(q)$	Grau de saída (*out-degree*) de um nó q em um grafo direcionado
$d^-(q)$	Grau de entrada (*in-degree*) de um nó q em um grafo direcionado
$d(q)$	Grau de um nó q em um grafo
η	Ruído gaussiano com média 0 e variância σ^2
ℓ	Comprimento de um caminho em um grafo
ρ	Comprimento de um ciclo em um grafo
α	Grau de repetição na entrada do codificador/decodificador RA/RA modificado
β	Grau de soma na saída do codificador/decodificador RA/RA modificado
$\lceil x \rceil$	Igual ao maior inteiro menor que x.
$\oplus$	Operador ou exclusivo (ou soma módulo 2)

CAPÍTULO 1

INTRODUÇÃO

A cada dia exigimos mais dos sistemas de comunicações. Tais exigências surgem de duas formas: primeiramente a necessidade de transmissão de maior quantidade de dados, a outra com o aumento do número de usuários dos sistemas de comunicações. Suprir essas exigências do mercado implica em maiores taxas de transmissão e maior capacidade nos sistemas de comunicações.

Com o constante progresso das telecomunicações há necessidade de se desenvolver novas ferramentas. A maior parte delas associada ao desenvolvimento de técnicas de processamento de sinais e ao surgimento de novos microprocessadores, cada vez mais velozes e menores.

Algumas descobertas teóricas foram fundamentais para o desenvolvimento e evolução dos sistemas de comunicações. Inicialmente, Claude E. Shannon em 1948 publicou uma série de resultados que se tornaram conhecidos como "A Teoria Matemática das Comunicações" [1], hoje conhecida como "A Teoria da Informação" [2], que mostrou que cada canal de comunicações tem uma capacidade limite característica que pode ser determinada. Exemplificando, cada linha telefônica permite transmitir dados até uma certa taxa de transmissão. Se precisarmos usar uma taxa maior teremos que usar um canal de maior capacidade limite. Se insistirmos em usar a linha telefônica com taxas maiores do que aquela limite, teremos uma transmissão com erros, comprometendo a informação transmitida. Com isso Shannon deu início às pesquisas na área de códigos corretores de erros. Um importante resultado gerado por Shannon foi a demonstração do fato de que para se chegar próximo da capacidade de canal um código de bloco teria um comprimento de palavra-código muito grande, teoricamente tendendo para o infinito. A dificuldade reside no fato de que os codificadores e (principalmente) os decodificadores para tais códigos são extremamente complexos.

O trabalho inicial para a obtenção das primeiras classes de bons códigos corretores de erros [3] foi árduo, pois exigia um profundo conhecimento de Álgebra Abstrata e Teoria de Probabilidade, sendo desenvolvido por um grupo restrito composto basicamente por matemáticos, embora a importância prática daquele tema já fosse reconhecida pelos

engenheiros de comunicações da época. Décadas se passaram e hoje se tem conhecimento de várias classes de bons códigos corretores de erros, que podem ser perfeitamente entendidas por engenheiros e cientistas de computação, graças ao esforço de vários pesquisadores, que souberam apresentar esse material de maneira clara e objetiva, sempre mantendo um bom rigor matemático.

1.1 MOTIVAÇÃO

Buscando aumentar as taxas das transmissões, diminuir a probabilidade de erros e obter o melhor desempenho possível utilizamos os códigos corretores de erros. Existem muitos tipos de códigos corretores de erros, e já foram realizados muitos avanços na área de codificação. Os critérios adotados no projeto de códigos variam segundo as características dos canais de comunicação utilizados, visando sempre o melhor desempenho de erro. Neste trabalho vamos utilizar tanto o canal Gaussiano (AWGN) como o canal de resposta parcial do tipo $1-D$ [23], e verificar o desempenho dos códigos de verificação de paridade de baixa densidade (LDPC, *Low Density Parity-Check Codes*) [3] nestes canais.

Os LDPC foram introduzidos junto com um algoritmo de decodificação iterativo, desenvolvido por Gallager no início da década de 60 [5]. Eles foram construídos usando matrizes de verificação de paridade esparsas e aleatórias, e devido à "esparsidade" estão associados a decodificadores de baixa complexidade. Os LDPC ficaram esquecidos por mais de três décadas até que, com o advento dos códigos turbo [3] (que até então formavam a classe de códigos com a qual se chegava mais próximo da capacidade de canal, porém com uma complexidade de decodificação ainda bastante elevada), em 2003 [3], foram "redescobertos" por MacKay [6], que demonstrou o potencial desses códigos para se alcançar a capacidade de canal, comparável ao dos códigos turbo. Mais recentemente, Richardson e Urbanke [7] desenvolveram os códigos LDPC irregulares, que possuem um desempenho melhor do que aquele dos códigos turbo para blocos com comprimentos muito grandes ($>10^5$), podendo chegar a até 0,1 dB da capacidade de Shannon.

1.2 OBJETIVOS

O principal objetivo desta dissertação de mestrado é o estudo dos códigos LDPC. Estudar suas características: como é feita a geração dos códigos, quais são as taxas utilizadas, o que é necessário para codificar e decodificar, e conhecer as vantagens em relação a outros códigos.

Também propor a utilização de códigos LDPC em sistemas de gravação magnética, criando um codificador e um decodificador para o uso em canais de resposta parcial. Realizar simulações com o canal do tipo $1-D$, e verificar se é viável ou não a proposta.

Hoje um número maior de sistemas tem utilizado os códigos LDPC. Entender o porquê desta preferência também é um dos objetivos. Os códigos LDPC são utilizados em sistemas bastante conhecidos, e sua adoção foi sugerida por um grupo de pesquisadores, que inclui professores e alunos do GPqCom, para o sistema brasileiro de televisão digital (SBTVD). Num outro projeto do mesmo grupo, os códigos LDPC deverão ser adotados num sistema do tipo WiMAX de transmissão sem fio em faixa larga.

1.3 CONTRIBUIÇÕES DA DISSERTAÇÃO

Foi proposto um método de codificação simples para código LDPC submetido ao canal $1-D$. O método de decodificação é o mesmo utilizado em vários trabalhos e utiliza o algoritmo soma-produto (SPA).

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação esta dividida em seis capítulos. O Capítulo 1 apresenta a introdução, onde são explicados o tema e o objetivo do trabalho. O Capítulo 2 versa sobre os códigos de bloco, onde é explicada a teoria necessária para o entendimento de códigos de bloco, que servem como passo inicial ao entendimento de códigos LDPC. O Capítulo 3 trata da teoria dos grafos, onde é apresentada a teoria dos grafos moldada de uma forma destinada a sua posterior aplicação em códigos LDPC. No Capítulo 4 é realizado o estudo dos códigos LDPC, codificação, decodificação e construção. No Capítulo 5 é apresentada a proposta de utilização

dos códigos LDPC em canais de gravação magnética. O Capítulo 6 apresenta a conclusão sobre o trabalho.

CAPÍTULO 2

CÓDIGOS DE BLOCO

Os códigos de bloco pertencem a uma classe muito importante de códigos corretores de erros. Os códigos LDPC podem ser considerados códigos de bloco, mas como veremos mais adiante, são representados por matrizes de paridade esparsas. Neste capítulo vamos explicar os processos de codificação e decodificação de códigos de bloco para servir como base para o entendimento dos códigos LDPC.

2.1 CARACTERÍSTICAS GERAIS E REPRESENTAÇÃO

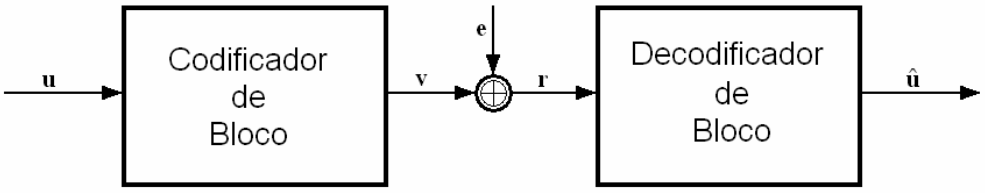


Figura 2.1: Sistema de comunicação binária com código de bloco

Considere uma fonte de informação tal que a seqüência produzida por esta fonte é subdividida em blocos de comprimento k cada. O codificador produzirá para cada bloco de informação um outro bloco, de comprimento n, com $n > k$. A diferença $n-k$ representa o número de símbolos redundantes utilizados na detecção e correção de erros. No caso de a fonte ser binária, o código é dito binário e os símbolos da fonte pertencem ao alfabeto binário $\{0,1\}$. Neste trabalho, somente serão considerados códigos binários, embora existam códigos de bloco não binários.

Na Figura 2.1 é apresentado um sistema de codificação e decodificação de códigos de bloco, $\mathbf{u}$ representa um bloco de comprimento k de informação, $\mathbf{v}$ é a palavra-código com comprimento n, $\mathbf{r}$ é a palavra-código $\mathbf{v}$ somada ao vetor erro $\mathbf{e}$, todos com comprimento n. O bloco de informação decodificado esta representado por $\hat{\mathbf{u}}$ e possui comprimento k. O vetor de erro $\mathbf{e}$ possui 1's em algumas posições e 0's nas demais. Quando o vetor erro é somado

(módulo 2) a palavra-código, provoca erros nas posições nas quais é igual a 1. Esses vetores $\mathbf{e}$ podem ser determinados e classificados, e recebem o nome de padrões de erros.

Os códigos de bloco podem ser descritos matematicamente através de conceitos de álgebra linear. Como vamos tratar apenas de códigos binários, utilizaremos a aritmética de módulo 2. Projetar um código de bloco nada mais é do que escolher 2^k seqüências de 0 e 1 , de comprimento n, chamadas de palavras-código, dentre as 2^n n-uplas (seqüências de comprimento n) binárias possíveis. Vamos definir então os códigos de bloco como um subconjunto **C** do conjunto de todas as 2^n n-uplas binárias possíveis.

Para definir um código de bloco de taxa $R = k/n$, precisamos escolher 2^k palavras-código. O nosso objetivo é reduzir os possíveis erros em uma transmissão de dados. Então temos que escolher as palavras-código pensando em minimizar a probabilidade de erro. No decorrer deste trabalho, trataremos melhor esse problema e utilizaremos alguns métodos de se realizar essa escolha.

Vamos definir agora a distância de Hamming entre duas palavras binárias como sendo a contagem dos bits com valores lógicos complementares em posições correspondentes nas duas palavras, ou seja, o número de posições em que as palavras binárias diferem.

2.2 DECODIFICAÇÃO DE MÁXIMA VEROSSIMILHANÇA

Vamos supor que $\mathbf{u}$ seja o vetor de informação e que a palavra-código **v** tenha sido transmitida. O decodificador deverá estimar $\hat{\mathbf{u}}$, ou $\hat{\mathbf{v}}$, a partir da observação $\mathbf{r}$. Dizemos que um erro de palavra-código acontece quando $\hat{\mathbf{u}} \neq \mathbf{u}$ ou, equivalentemente, quando $\hat{\mathbf{v}} \neq \mathbf{v}$. Denotemos por E o evento erro de palavra-código. Assim, a probabilidade de erro de palavra-código, dado que o vetor recebido foi $\mathbf{r}$, é dada por:

$$P(E \mid \mathbf{r}) \equiv P(\hat{\mathbf{v}} \neq \mathbf{v} \mid \mathbf{r}). \tag{2.1}$$

A probabilidade média de erro de palavra-código é dada por:

$$P(E) = \sum_{\mathbf{r}} P(E \mid \mathbf{r}) P(\mathbf{r}). \tag{2.2}$$

O decodificador deverá se empenhar para que a estimativa $\hat{\mathbf{v}}$ seja aquela que minimize a probabilidade condicionada de erro $P(E|\mathbf{r})$. Mas, minimizar $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ é equivalente a maximizar a probabilidade $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$. Usando a regra de Bayes, podemos escrever a probabilidade $P(\mathbf{v}|\mathbf{r})$ como:

$$P(\mathbf{v}|\mathbf{r}) = \frac{P(\mathbf{v},\mathbf{r})}{P(\mathbf{r})} = \frac{P(\mathbf{r}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})}. \tag{2.3}$$

Supondo que os 2^k vetores de informação sejam eqüiprováveis, tem-se que as 2^k palavras-código também são eqüiprováveis: $P(\mathbf{v}) = \frac{1}{2^k}$, ou seja, $P(\mathbf{v})$ não depende de $\mathbf{v}$. Por outro lado, $P(\mathbf{r})$ também não depende de **v**. Assim, maximizar a Equação 2.3 é equivalente a maximizar $P(\mathbf{r}|\mathbf{v})$. Esta é conhecida como função de verossimilhança, porque é uma medida de quão verossímil (ou parecido com o verdadeiro) é o vetor $\mathbf{r}$. Como estamos considerando um canal sem memória, o bit r_i (onde $i = \{1, 2, ..., n\}$) depende apenas do bit v_i, portanto não depende estatisticamente do bit transmitido v_j, para $i \neq j$. Temos que maximizar a seguinte expressão:

$$P(\mathbf{r}|\mathbf{v}) = P\left[(r_1, r_2, ..., r_n)\middle|(v_1, v_2, ..., v_n)\right] = \prod_{i=1}^{n} P(r_i|v_i). \tag{2.4}$$

Para o canal AWGN com transmissão binária bipolar $(0 \rightarrow \text{-}1 \text{ e } 1 \rightarrow +1)$, temos:

$$P(r_i|v_i) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(r_i+1)^2}{2\sigma^2}}, \text{para } v_i = 0 \\ \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(r_i-1)^2}{2\sigma^2}}, \text{para } v_i = 1 \end{cases}$$

$$P(\mathbf{r}|\mathbf{v}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(r_i-v_i)^2}{2\sigma^2}} = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^n} e^{-\sum_{i=1}^{n}\frac{(r_i-v_i)^2}{2\sigma^2}}. \tag{2.5}$$

Claramente, a palavra-código $\hat{\mathbf{v}}$ que maximiza $P(\mathbf{r} \mid \mathbf{v})$ é aquela que minimiza a distância Euclidiana quadrática entre $\hat{\mathbf{v}}$ e $\mathbf{r}$, definida por:

$$d_E^2(\hat{\mathbf{v}}, \mathbf{r}) = \sum_{i=1}^{n} (r_i - \hat{v}_i)^2 . \tag{2.6}$$

Isso significa que o critério de máxima verossimilhança é equivalente nesse caso ao critério de mínima distância Euclidiana quadrática. O decodificador de máxima verossimilhança é de simples implementação, entretanto a quantidade de cálculos necessários para se poder tomar a decisão é muito grande, pois é necessário calcular uma distância Euclidiana para cada palavra-código. Assim, a decodificação de máxima verossimilhança, nessa versão "força-bruta", só pode ser adotada para códigos pequenos. No entanto, propriedades algébricas podem ser impostas aos códigos, resultando em procedimentos de decodificação de máxima verossimilhança bem menos complexos.

2.3 CÓDIGOS DE BLOCO LINEARES

Os códigos de bloco lineares são uma classe muito importante de códigos de detecção e correção de erros. Quando a soma módulo 2 de quaisquer duas palavras-código resulta em outra palavra-código, isto é, se para quaisquer palavras-código $\mathbf{v} \in \mathbf{C}$ e $\mathbf{v}' \in \mathbf{C}$ tivermos que $\mathbf{v} \oplus \mathbf{v}' = \mathbf{v}'' \in \mathbf{C}$, classifica-se o código como linear. Como a soma módulo 2 de uma palavra-código com ela mesma resulta no vetor nulo, este é palavra-código de todo o código linear.

Como exemplo de um código linear, temos o código de bloco de Hamming estendido, de taxa $R = 4/8$.

u					v							
0	0	0	0		0	0	0	0	0	0	0	0
0	0	0	1		0	1	1	1	0	0	0	1
0	0	1	0		1	0	1	1	0	0	1	0
0	0	1	1		1	1	0	0	0	0	1	1
0	1	0	0		1	1	0	1	0	1	0	0
0	1	0	1		1	0	1	0	0	1	0	1
0	1	1	0		0	1	1	0	0	1	1	0
0	1	1	1		0	0	0	1	0	1	1	1
1	0	0	0	⇒	1	1	1	0	1	0	0	0
1	0	0	1		1	0	0	1	1	0	0	1
1	0	1	0		0	1	0	1	1	0	1	0
1	0	1	1		0	0	1	0	1	0	1	1
1	1	0	0		0	0	1	1	1	1	0	0
1	1	0	1		0	1	0	0	1	1	0	1
1	1	1	0		1	0	0	0	1	1	1	0
1	1	1	1		1	1	1	1	1	1	1	1

Tabela 2.1: Código de Hamming estendido de Taxa *R*=4/8

2.3.1 MATRIZ GERADORA

Dado um código de bloco linear com taxa $R = k/n$, ou seja, as palavras-código $\mathbf{v} = (v_1, v_2, ..., v_n)$ têm dimensão n e a informação a ser codificada é dividida em blocos $\mathbf{u} = (u_1, u_2, ..., u_k)$ de dimensão k, sendo $n > k$. A forma mais simples de se realizar esta codificação é através de uma matriz $\mathbf{G}$ de dimensões $k \times n$. A palavra-código pode ser obtida através da seguinte equação matricial:

$$\mathbf{v} = \mathbf{uG} . \tag{2.7}$$

Essa matriz recebe o nome de matriz geradora, porque as palavras-código são obtidas através do produto da informação com a matriz $\mathbf{G}$.

Concluímos facilmente, a partir da Equação 2.7, que cada bit da palavra-código é uma combinação linear dos bits de informação. Para constituirmos um código útil, temos que garantir que os vetores de informação, originem palavras-código distintas, possibilitando sua

posterior decodificação. Para isso as linhas da matriz **G** devem ser necessariamente linearmente independentes, ou seja, a matriz **G** deve ser uma matriz de posto completo (*full rank*).

No caso de o código ser sistemático, as palavras-código são formadas pelos próprios bits de informação e pelos bits de verificação de paridade associados:

$$\mathbf{v} = (b_1, b_2, \ldots, b_{n-k}, u_1, u_2, \ldots, u_k)$$

onde os bits b_i, com $i = \{1, 2, \ldots, n-k\}$, são os bits de verificação de paridade, e os bits u_i, com $i = \{1, 2, \ldots, k\}$, são os bits de informação. Neste caso os bits da palavra-código podem ser obtidos de forma separada: $\mathbf{b} = \mathbf{uP}$, sendo $\mathbf{P}$ uma matriz de dimensão $k \times (n-k)$, e a matriz geradora é construída da seguinte forma: $\mathbf{G} = [\mathbf{P} \mid \mathbf{I_k}]$, sendo $\mathbf{I_k}$ a matriz identidade de dimensão $k \times k$.

Exemplo 2.1: A matriz geradora para o código de Hamming de taxa 4/8, correspondente à Tabela 2.1 é:

$$\mathbf{G} = \left[\begin{array}{cccc:cccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \Rightarrow \mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

2.3.2 MATRIZ DE VERIFICAÇÃO DE PARIDADE

As 2^k palavras-código de um código binário linear satisfazem um conjunto de $n-k$ equações lineares homogêneas em v_i com $i = \{1, 2, \ldots, n\}$, isto é, um sistema de $n-k$ equações homogêneas com n incógnitas. Estas equações são chamadas de **equações de verificação de paridade**, e podem ser escritas na forma matricial como:

$$\mathbf{H}\mathbf{v}^{\mathbf{T}} = 0_{(n-k)\times 1}, \tag{2.8}$$

em que $\mathbf{H}$ é a matriz de verificação de paridade, de dimensões $(n-k)\times n$, formada pelos coeficientes do sistema de $n-k$ equações homogêneas com n incógnitas. As mesmas regras anteriormente aplicadas à matriz $\mathbf{G}$ são aplicadas à matriz $\mathbf{H}$, portanto as linhas da matriz $\mathbf{H}$ devem ser linearmente independentes. Caso as linhas de $\mathbf{H}$ não sejam linearmente independentes, teremos um código de taxa $R = k'/n$, sendo $k' > k$ (por eliminação das linhas linearmente dependentes).

Das Equações 2.7 e 2.8 temos:

$$\mathbf{H}\mathbf{v}^{\mathbf{T}} = 0_{(n-k)\times 1} \Leftrightarrow \mathbf{v}\mathbf{H}^{\mathbf{T}} = 0_{1\times(n-k)} \Leftrightarrow \mathbf{u}\mathbf{G}\mathbf{H}^{\mathbf{T}} = 0_{1\times(n-k)} \Rightarrow \mathbf{G}\mathbf{H}^{\mathbf{T}} = 0_{k\times(n-k)}, \tag{2.9}$$

Um código de bloco linear definido por uma matriz $\mathbf{H}$ de posto completo (*full rank*) é sempre sistematizável e, nesse caso, realizando operações lineares nas linhas de $\mathbf{H}$ podemos escrever a matriz da forma sistemática: $\mathbf{H} = \left[\mathbf{I}_{n-k} | \mathbf{P}^{\mathbf{T}}\right]$, onde $\mathbf{I}_{n-k}$ é a matriz identidade de dimensões $(n-k)\times(n-k)$.

2.4 CAPACIDADE DE DETECÇÃO E CORREÇÃO DE ERROS

Para determinarmos a capacidade de detecção ou de correção de erros de códigos de bloco, precisamos conhecer alguns conceitos. A distância de Hamming entre dois vetores **a** e **b**, denotada por $d_H(\mathbf{a},\mathbf{b})$, é definida como o número de posições em que **a** e **b** diferem. O peso de Hamming de um vetor **a**, denotado por $\omega_H(\mathbf{a})$, é definido como o número de coordenadas não nulas do vetor **a**.

Exemplo 2.2: Dados os vetores $\mathbf{a} = [1\ 1\ 0\ 1\ 0\ 1\ 0]$ e $\mathbf{b} = [0\ 0\ 1\ 0\ 1\ 0\ 1]$, temos:

- Distância de Hamming: $d_H(\mathbf{a},\mathbf{b}) = 7$
- Peso de Hamming: $\omega_H(\mathbf{a}) = 4$, $\omega_H(\mathbf{b}) = 3$

A distância mínima de um código **C**, denotada por $d_{\min}(\mathbf{C})$, é a menor distância de Hamming verificada para todos os pares distintos de palavras-código pertencentes ao código **C**. Como usamos aritmética módulo 2, para duas palavras-código $\mathbf{a} \in \mathbf{C}$ e $\mathbf{b} \in \mathbf{C}$, onde **C** é linear, temos que:

$$d_H(\mathbf{a},\mathbf{b}) = \omega_H(\mathbf{a}+\mathbf{b}) = \omega_H(\mathbf{c}),\ \mathbf{c} \in \mathbf{C}. \tag{2.10}$$

Assim, a distância mínima de um código linear é igual ao peso mínimo excluindo a palavra-código toda nula.

Agora dado um código de bloco de taxa $R = k/n$ e cuja distância mínima é $d_{\min}$, queremos determinar quantos erros de bits ele é capaz de detectar, e quantos pode corrigir.

2.4.1 A CAPACIDADE DE DETECÇÃO

Um código tem capacidade de detecção de N_D erros de bit se ele for capaz de detectar qualquer padrão de erro **e** tal que $\omega_H(\mathbf{e}) \le N_D$, e houver pelo menos um padrão de erro **e** com $\omega_H(\mathbf{e}) = N_D + 1$ tal que o código não seja capaz de detectar.

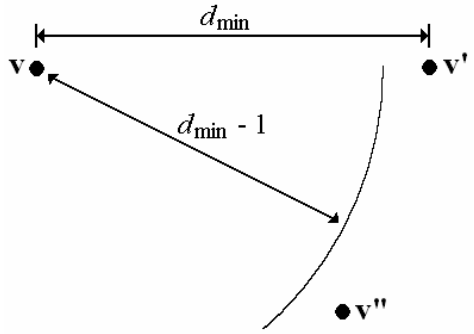


Figura 2.2: Espaço contendo três palavras-código de um código de Bloco.

Na Figura 2.2, considere as palavras-código **v** e **v'**, distantes uma da outra de $d_H(\mathbf{v},\mathbf{v}') = d_{\min}$, e uma terceira palavra-código **v"**, mais afastada, como pontos no espaço. Devemos notar que, se o vetor erro **e** tiver peso de Hamming $\omega_H(\mathbf{e}) \le d_{\min} - 1$, então a palavra-código transmitida **v** estará a uma distância menor ou igual a $d_{\min} - 1$ do vetor recebido $\mathbf{r} = \mathbf{v} \oplus \mathbf{e}$. Isto significa que **r** não pode ser uma palavra-código. Portanto, podemos afirmar que o código é capaz de detectar todos os padrões de erro cujo peso da Hamming seja $\le d_{\min} - 1$.

Exemplo 2.3: Para o código de Hamming de taxa 4/8, que é apresentado na Tabela 2.1, temos: $d_{\min} = 4$, como a capacidade de correção de erros é igual a $d_{\min} - 1$ o código é capaz de detectar 3 erros.

2.4.2 A CAPACIDADE DE CORREÇÃO

Um código tem capacidade de correção de N_C erros de bit se ele for capaz de corrigir qualquer padrão de erro **e** tal que $\omega_H(\mathbf{e}) \le N_C$, e houver pelo menos um padrão de erro **e** com $\omega_H(\mathbf{e}) = N_C + 1$ que o código não seja capaz de corrigir.

Quando consideramos a correção de um erro é necessário levar em conta que, diferentemente da detecção do erro, temos que transformar a palavra **r** recebida em uma palavra-código.

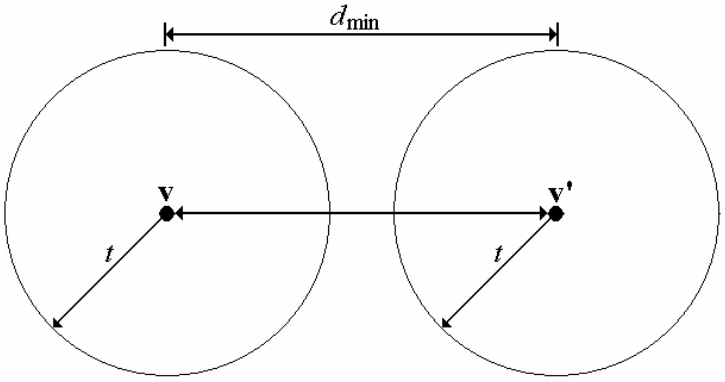


Figura 2.3: Espaço ilustrando a capacidade de correção de erro de um código de Bloco.

Considere a Figura 2.3, onde são mostradas duas palavras-código **v** e **v'**, distantes uma da outra de $d_H(\mathbf{v},\mathbf{v}') = d_{\min}$. Vamos agora imaginar 2^k esferas de raio *t* soltas no espaço, com centros exatamente nos pontos correspondentes às palavras-código. Vamos supor que a palavra-código **v**, que é o centro da esfera E_v, tenha sido transmitida. O decodificador usará a seguinte regra de decodificação:

1. Se o vetor recebido **r** estiver dentro ou na superfície da esfera E_v, então a palavra-código correspondente ao centro desta esfera (ou seja, **v**) será a palavra decodificada (dada como correta). Com esta regra, se o número de erros for $\leq t$, o decodificador será capaz de corrigir o(s) erro(s) com sucesso.
2. Se o vetor recebido **r** estiver dentro de uma outra esfera, digamos $E_{v'}$, então a palavra decodificada será $\mathbf{v}'$, e a decodificação será errônea.
3. Se o vetor recebido **r** estiver fora de qualquer esfera, o decodificador poderá decidir-se por uma das palavras-código mais próximas, ou declarar apenas que houve erro na transmissão.

Quanto maior o valor de *t*, maior será o número de erros que o código será capaz de corrigir. As esferas não podem se tocar, se isto acontecer teremos um ponto de indecisão no decodificador e isso não é aceitável. Assim, $d_{\min} \geq 2t+1$. A capacidade de correção do código pode ser obtida através da seguinte formula:

$$t = \left\lceil \frac{d_{\min} - 1}{2} \right\rceil, \tag{2.11}$$

onde t corresponde ao número de erros que o código é capaz de corrigir, e $\lceil x \rceil$ denota o maior inteiro menor que x.

Exemplo 2.4: Para o código de Hamming de taxa 4/8, da Tabela 2.1, temos:

$d_{\min} = 4$, portanto: $t = \left\lceil \frac{d_{\min} - 1}{2} \right\rceil = \left\lceil \frac{4-1}{2} \right\rceil = 1,5$ o código é capaz de corrigir 1 erro.

2.5 DECODIFICAÇÃO UTILIZANDO O MÉTODO DA SÍNDROME

Um dos métodos mais eficientes na decodificação de códigos de bloco lineares de taxa $R = k/n$, para baixos valores de n, consiste na utilização do conceito de síndrome. Considere que ao transmitirmos a palavra-código **v**, o vetor recebido seja:

$$\mathbf{r} = \mathbf{v} \oplus \mathbf{e}, \tag{2.12}$$

em que **e** é o padrão de erro introduzido pelo canal de transmissão. A primeira etapa da decodificação consiste em verificar se a palavra recebida é uma palavra-código. Podemos fazer esta verificação utilizando a Equação 2.8, mas com **r** no lugar de **v**. Então temos:

$$\mathbf{H}\mathbf{r}^{\mathbf{T}} = \mathbf{s}_{(\mathbf{n-k})\times\mathbf{1}} \Leftrightarrow \mathbf{r}\mathbf{H}^{\mathbf{T}} = \mathbf{s}_{\mathbf{1}\times(\mathbf{n-k})}, \tag{2.13}$$

onde o vetor **s** é conhecido como vetor de síndrome. Quando ele é igual ao vetor nulo, o vetor recebido é uma palavra-código, e a decodificação é imediata. No caso de um código sistemático, eliminando os $n-k$ bits de paridade temos a informação. Para o caso dos códigos não sistemáticos, a forma mais simples é dispor de uma tabela em memória que faça a correspondência entre a palavra-código e a informação.

Se $\mathbf{s} \neq 0$, o que significa que **r** não corresponde a uma palavra-código, então devemos corrigir os erros que foram provocados pelo canal. Isso pode ser feito utilizando o próprio vetor de síndrome. Utilizando as Equações 2.12 e 2.13, temos:

$$\mathbf{r} = \mathbf{v} \oplus \mathbf{e}, \ \mathbf{s} = \mathbf{r}\mathbf{H}^{\mathbf{T}} \tag{2.14}$$

$$\mathbf{s} = (\mathbf{v} + \mathbf{e})\mathbf{H}^{\mathbf{T}} \Leftrightarrow \mathbf{s} = \mathbf{v}\mathbf{H}^{\mathbf{T}} + \mathbf{e}\mathbf{H}^{\mathbf{T}} \Leftrightarrow \mathbf{s} = \mathbf{0} + \mathbf{e}\mathbf{H}^{\mathbf{T}} \Leftrightarrow \mathbf{s} = \mathbf{e}\mathbf{H}^{\mathbf{T}} \tag{2.15}$$

Através da Equação 2.15, podemos concluir que o vetor síndrome depende exclusivamente do padrão de erro **e**. Ele é obtido pela soma das colunas da matriz **H** correspondente aos bits errados, portanto, é dado pelas posições dos bits 1 do padrão de erro. Existem 2^{n-k} síndromes distintas cada uma delas associada a um padrão de erro. Devido às características estatísticas do canal binário simétrico, os 2^{n-k} padrões de erros escolhidos são os que possuem o menor número de bits 1, ou seja os de menores peso de Hamming. No caso de existir apenas um vetor de erro correspondente, podemos corrigir a palavra recebida. Caso contrário, apenas poderemos detectar os erros de transmissão, mas não poderemos corrigi-los. Por existir mais de uma possibilidade para esta correção não podemos ter certeza de qual delas é a correta.

O algoritmo para a decodificação de uma palavra recebida é:

i. Cálculo da síndrome: $\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathbf{T}}$;

ii. Determinação do vetor de erro **e** associado à síndrome; a forma mais simples de se fazer isso é possuir uma tabela em uma memória com os respectivos vetores de erro associados às síndromes;

iii. Determinação da palavra-código transmitida: $\mathbf{v} = \mathbf{r} \oplus \mathbf{e}$;

iv. Determinação da informação transmitida.

2.6 CONCLUSÕES DO CAPÍTULO

Neste capítulo estudamos os códigos de bloco, suas propriedades, codificação e decodificação. No próximo capítulo vamos estudar a teoria dos grafos. Estes dois capítulos servirão como base para o entendimento dos códigos LDPC. Códigos LDPC também são

códigos de bloco, mas, como veremos nos próximos capítulos deste trabalho, possuem algumas propriedades específicas.

CAPÍTULO 3

TEORIA DOS GRAFOS

Estruturas que podem ser representadas por grafos estão em toda parte e muitos problemas de interesse prático podem ser formulados como questões sobre certos grafos. O primeiro trabalho científico sobre grafos foi escrito em 1736 pelo matemático suíço Leonhard Euler a respeito do problema das pontes da cidade de Königsberg. O problema consistia em atravessar todas as sete pontes que conectavam a cidade sem passar duas vezes pela mesma ponte.

3.1 CARACTERÍSTICAS GERAIS E REPRESENTAÇÃO

Um grafo é um conjunto de pontos, chamados nós (ou vértices), conectados por linhas, chamadas de arestas. As arestas podem ligar um nó a um outro nó ou a ele mesmo; também é possível associar um peso numérico à aresta conforme a necessidade da aplicação. Quando as arestas têm uma direção definida, indicamos graficamente através de uma seta, e teremos um grafo direcionado, ou dígrafo.

Para entendermos melhor o que é um grafo, vamos defini-los matematicamente. Seja o grafo $\mathbf{D(V, A)}$, onde **V** é um conjunto finito e não vazio, e **A** é um conjunto de pares não ordenados de elementos de **V**. Considere o grafo $\mathbf{D(V, A)}$ como:

$$\mathbf{V} = \{1, 2, 3, 4, 5\}$$
$$\mathbf{A} = \{(1,2), (1,4), (2,3), (2,5), (3,4)\}$$

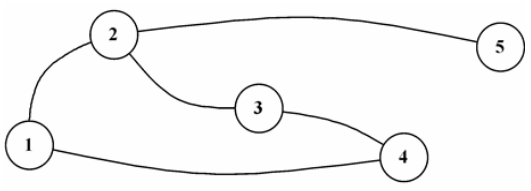


Figura 3.1: Representação gráfica de um grafo.

Portanto **V** representa os nós e **A** representa o conjunto das arestas. Cada aresta **a** pertencente ao conjunto **A** será denotada pelo par de nós que a forma: $\mathbf{a} = (q, w)$. Os nós q e w

são os extremos da aresta e são denominados adjacentes. A aresta **a** é dita incidente a ambos os nós q e w. A Figura 3.1 é um exemplo de uma representação gráfica de um grafo.

3.2 CLASSIFICAÇÕES DE GRAFOS

Os grafos são divididos em várias classes, dependendo de algumas características tais como: se um nó se liga a ele mesmo, se as arestas são direcionadas ou não, se as arestas possuem pesos ou não, entre outras características que serão apresentadas neste capítulo. Lembrando que estaremos direcionando o estudo dos grafos à nossa aplicação neste trabalho.

3.2.1 GRAFOS DIRECIONADOS E NÃO DIRECIONADOS

Seja um grafo direcionado **D(V, A)**, onde **V** é um conjunto finito e não vazio de nós e **A** é uma relação binária em **V**. Uma aresta $\mathbf{a} = (q, w)$ sai do nó q e entra no nó w. Portanto, o nó q é adjacente ao nó w. Podem existir arestas de um nó para ele mesmo, chamadas de *self-loops*. Um exemplo é mostrado na Figura 3.2.a.

Seja um grafo não direcionado **D(V, A)**, onde **A** é um conjunto de pares não ordenados de elementos distintos de **V**. As arestas $\mathbf{a_1} = (q, w)$ e $\mathbf{a}_2 = (w, q)$ são consideradas como uma única aresta. A relação de adjacência é simétrica. Não podem existir *self-loops*. Um exemplo é mostrado na Figura 3.2.b.

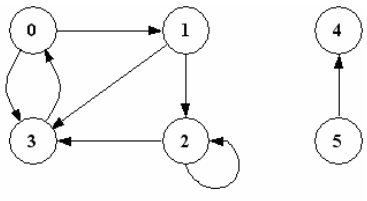


Figura 3.2.a: Exemplo de um grafo direcionado.

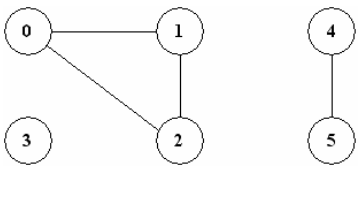


Figura 3.2.b: Exemplo de um grafo não direcionado.

3.2.2 GRAU DE UM NÓ

Quando estamos trabalhando com nós pertencentes a grafos não direcionados, o grau de um nó é o número de arestas que incidem sobre ele. Entretanto, quando os nós pertencem a grafos direcionados, o grau é o número de arestas que saem dele (*out-degree*) mais o número de arestas que chegam nele (*in-degree*). Para todos os grafos um nó de grau zero é dito isolado ou não conectado.

O número de nós de um grafo direcionado **D** é definido como a ordem deste grafo e o número de arestas em **D** como o seu tamanho. Se $\mathbf{a} = (q, w)$ é uma aresta de **D** então *q* é dito ser adjacente de *w* e *w* é adjacente de *q*. Além disso, a aresta $\mathbf{a} = (q, w)$ é convergente em *w* e divergente em q. O grau de saída (*out-degree*), $d^+(q)$, referente a um nó *q* em um grafo direcionado **D** é o número de arestas divergentes em q. O grau de entrada (*in-degree*), $d^-(q)$, é o número de arestas convergente em q. O grau $d(q)$ de um nó *q* de um grafo direcionado **D** é definido como $d(q) = d^+(q) + d^-(q)$.

Exemplo 3.1: Podemos determinar os graus no grafo da Figura 3.2.a, de acordo com a tabela abaixo:

Nó	Grau de saída	Grau de entrada	Grau
0	2	1	3
1	2	1	3
2	2	2	4
3	1	3	4
4	0	1	1
5	1	0	1

Para o grafo da Figura 3.2.b, temos, por exemplo, o nó 1 tem grau 2 e o nó 3 é isolado.

Seja **D** um grafo direcionado de ordem *p* e tamanho *m*, com $\mathbf{V} = \{q_1, q_2, ..., q_{p-1}, q_p\}$. Então, da analise acima podemos concluir que:

$$\sum_{i=1}^{p} d^{+}(q_i) = \sum_{i=1}^{p} d^{-}(q_i) = m .$$

3.2.3 CAMINHO ENTRE NÓS

Um caminho de comprimento ℓ de um nó *x* a um nó *y* em um grafo **D(V, A)** é uma seqüência de nós $\mathbf{V} = \{q_0, q_1, \ldots, q_\ell\}$ tal que $x = q_0$ e $y = q_\ell$, e $(q_{i-1}, q_i) \in \mathbf{A}$ para $i = \{1, 2, \ldots, \ell\}$.

O comprimento de um caminho é o número de arestas percorridas nele, o caminho contém os nós $\mathbf{V} = \{q_0, q_1, \ldots, q_\ell\}$ e as arestas $\mathbf{A} = \{(q_0, q_1), (q_1, q_2), \ldots, (q_{\ell-1}, q_\ell)\}$. Se existir um caminho **Z** de *x* a *y*, então *y* é alcançável a partir de *x* via **Z**. Um caminho é simples se todos os nós do caminho são distintos.

Exemplo 3.2: Para o grafo da Figura 3.2.a temos:

O caminho $(0,1,2,3)$ é simples e tem comprimento 3. O caminho $(1,3,0,3)$ não é simples.

3.2.4 ISOMORFISMO

Um isomorfismo entre dois grafos **D** e **J** é uma bijeção f de $\mathbf{V}(\mathbf{D})$ em $\mathbf{V}(\mathbf{J})$ tal que dois nós *q* e *w* são adjacentes em **D** se e somente se $f(q)$ e $f(w)$ são adjacentes em **J**.

Dois grafos **D** e **J** são isomorfos se existe um isomorfismo entre eles. Em outras palavras, dois grafos são isomorfos se é possível alterar os nomes dos nós de um deles de tal modo que os dois grafos fiquem iguais. Para decidir se dois grafos **D** e **J** são isomorfos, basta examinar todas as bijeções de $\mathbf{V}(\mathbf{D})$ em $\mathbf{V}(\mathbf{J})$.

3.2.5 CICLOS EM GRAFOS DIRECIONADOS E NÃO DIRECIONADOS

Em um grafo direcionado, um caminho $\{q_0, q_1, \ldots, q_\ell\}$ forma um ciclo se $q_0 = q_\ell$ e o caminho contém pelo menos uma aresta. O ciclo é simples se os nós $\{q_0, q_1, \ldots, q_\ell\}$ são distintos. O *self-loop* é um ciclo de comprimento 1. Dois caminhos $\{q_0, q_1, \ldots, q_\ell\}$ e $\{q'_0, q'_1, \ldots, q'_\ell\}$ formam o mesmo ciclo se existir um inteiro *j* tal que $v'_i = v_{(i+j) \bmod \ell}$ para $i = \{0,1,\ldots,\ell\}$.

Em um grafo não direcionado um caminho $\{q_0, q_1, \ldots, q_\ell\}$ forma um ciclo se $q_0 = q_\ell$ e o caminho contém pelo menos três arestas. O ciclo é simples se os nós $\{q_1, q_2, \ldots, q_\ell\}$ são distintos.

Exemplo 3.3: Para o grafo da Figura 3.2.a temos:

O caminho $(0,1,2,3,0)$ forma um ciclo. O caminho $(0,1,3,0)$ forma o mesmo ciclo que os caminhos $(1,3,0,1)$ e $(3,0,1,3)$.

Para o grafo da Figura 3.2.b temos:

O caminho $(0,1,2,0)$ é um ciclo.

3.3 GRAFOS BIPARTIDOS

Um grafo é bipartido se existe uma bipartição do seu conjunto de nós tal que cada aresta tem uma ponta em uma das partes da bipartição e a outra ponta na outra parte. Em outras palavras, um grafo é bipartido se for possível atribuir uma de duas cores a cada nó de tal forma que as pontas de cada aresta tenham cores diferentes.

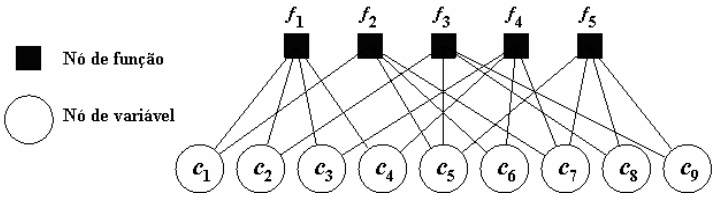
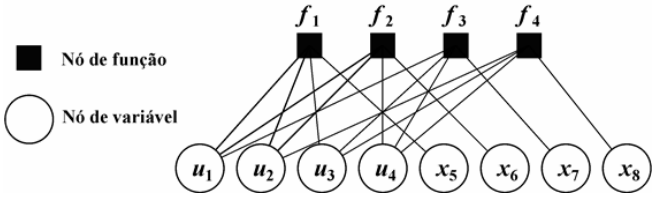


Figura 3.3: Exemplo de um grafo bipartido.

Na Figura 3.3, podemos observar dois grupos bem distintos de nós: o grupo de nós de função e o outro de nós de variável. Estes nomes foram dados por se tratar da representação de um código LDPC através de grafo, como veremos no próximo capítulo.

3.3.1 MATRIZ DE ADJACÊNCIA

Considere um grafo bipartido. É possível representar esse grafo através de uma matriz, que recebe o nome de matriz de adjacência. No caso de um código LDPC ou código de bloco, essa matriz recebe o nome de matriz de verificação de paridade, denotada por **H**. Como os códigos LDPC podem ser sempre representados por grafos bipartidos, podemos concluir que essas matrizes quando construídas de forma ordenada são equivalentes.



A partir do grafo acima vamos criar sua matriz adjacência, considere os nós f_1, f_2, f_3 e f_4 como sendo associados às linhas dessa matriz, e os nós, x_5, x_6, x_7, x_8, u_1, u_2, u_3 e u_4 como sendo associados às colunas. Quando houver uma aresta de conexão entre os nós se coloca o 1 indicando esta conexão, quando não houver conexão, se coloca 0. Portanto, temos:

	x_5	x_6	x_7	x_8	u_1	u_2	u_3	u_4
f_1	1	0	0	0	1	1	1	0
f_2	0	1	0	0	1	1	0	1
f_3	0	0	1	0	1	0	1	1
f_4	0	0	0	1	0	1	1	1

Podemos observar que a matriz de adjacência para o grafo é equivalente à matriz **H** do código de bloco apresentado no Capítulo 2. A matriz de verificação de paridade do código de Hamming de taxa 4/8 é apresentada abaixo:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

3.4 CONCLUSÕES DO CAPÍTULO

Neste capítulo apresentamos os conceitos básicos sobre a teoria dos grafos, suas características gerais, representações e classificações. No próximo capítulo, tomando como base os capítulos anteriores, vamos estudar os códigos LDPC.

CAPÍTULO 4

CÓDIGOS LDPC

Neste capítulo, iniciaremos o estudo dos códigos LDPC (*Low-density parity-check Code*). Um código LDPC é caracterizado por uma matriz de verificação de paridade esparsa, isto é, com uma baixa densidade de 1's. No que se refere à topologia dos códigos LDPC**,** é comum fazer-se a distinção entre códigos regulares e irregulares.

4.1 CÓDIGOS LDPC REGULARES E IRREGULARES

Os códigos LDPC regulares foram propostos em 1962 por Robert Gallager em sua tese de doutorado [8]. Um código LDPC regular é definido como sendo um código de bloco linear (n, k, w_c), com $w_c << m = n - k$, cuja matriz de verificação de paridade **H** tem dimensões $(n-k) \times n$, contendo w_c 1's por coluna e $w_r = w_c \times \frac{n}{m}$ 1's por linha.

Gallager demonstrou que fazendo $w_c \geq 3$, os conjuntos de códigos LDPC (n, k, w_c) que podem ser obtidos, na sua grande maioria, possuem uma distância mínima elevada. Mas para isso é necessário seguir algumas regras de construção. Por exemplo, duas colunas da matriz **H** devem possuir no máximo um 1 na mesma posição. Como o menor número de colunas de **H** que somadas resultam no vetor nulo correspondente à distância mínima do código, concluímos que o código LDPC possui uma distância mínima elevada.

Os códigos LDPC, mesmo com suas excelentes características, foram esquecidos pela comunidade científica, salvo raras exceções, até meados dos anos 90, quando foram redescobertos por Mackay e Neal [6], [9], [10]. Mackay e Neal demonstraram que os códigos LDPC conseguem atingir uma probabilidade de erro muito próxima ao limite estabelecido por Shannon [1]. O que levou o seu esquecimento por tantos anos foi a elevada complexidade computacional requerida tanto para a geração de uma matriz **H** que garanta uma boa distância mínima do código, como para sua codificação e decodificação.

Mais recentemente, surgiram os códigos LDPC irregulares, para os quais a matriz **H** possui uma baixa densidade de 1's mas o número de 1's por coluna e por linha não é constante. Em [7], pode ser encontrada uma demonstração de que os códigos LDPC

irregulares são superiores aos regulares para blocos longos. Entretanto, a implementação em hardware dos códigos LDPC regulares é mais simples.

4.1.2 GRAFOS DE TANNER

Um importante trabalho sobre códigos LDPC realizado durante o seu esquecimento pós-Gallager, e antes do seu ressurgimento, foi o estudo de Michael Tanner [11]. Tanner considerou que qualquer código linear de bloco, em particular códigos LDPC, podia ser representado por grafos bipartidos. Grafos bipartidos são formados por dois conjuntos de nós, onde os nós de um mesmo conjunto não se ligam entre si, mas sim aos nós do outro conjunto.

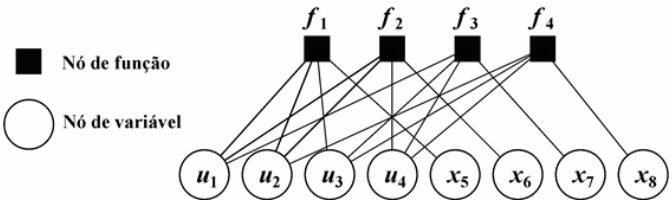


Figura 4.1: Exemplo de uma representação por grafo.

O código representado através do grafo da Figura 4.1 é o código de bloco de Hamming estendido de taxa $R = 4/8$, o mesmo utilizado como exemplo no Capítulo 2. A formação de palavras-código a partir de um grafo é bastante simples, como mostra o exemplo seguinte.

Exemplo 4.1: Construindo palavras-código a partir do grafo da Figura 4.1:

Seja o vetor de informação $\mathbf{u} = [0 \quad 1 \quad 0 \quad 1]$. Para obter as paridades referentes a esse vetor de informação vamos recorrer às propriedades do grafo. Em um nó de função a soma das informações enviadas sempre deve ser 0. Por exemplo, no nó f_1, temos $u_1 + u_2 + u_3 + x_5 = 0$.

Para o vetor de informação dado temos que $0+1+0+x_5 = 0 \therefore x_5 = 1$. Fazendo o mesmo para os demais nós de função temos: $x_6 = 0$, $x_7 = 1$, $x_8 = 0$.

Portanto a palavra-código é: $\mathbf{v} = [1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

A propriedade de a soma das informações enviadas dos nós de variável aos nós de função ser sempre zero módulo 2, possibilita a geração de palavras-código com grande eficiência e baixa complexidade.

4.1.3 CONCEITO DE *GIRTH*

Um dos conceitos mais importantes relativos aos grafos de Tanner é o de ciclo com comprimento ρ, definido como sendo um percurso fechado formado por ρ caminhos [13], [14]. Tendo como base a Figura 4.2, podemos observar um ciclo de comprimento 4 formado pelas arestas em negrito. O menor comprimento de todos os ciclos existentes num grafo de Tanner é designado por *girth*.

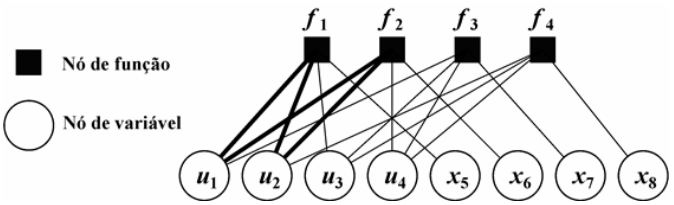


Figura 4.2: Exemplo de um *girth* em um grafo.

Na prática, ao projetar um código LDPC procura-se evitar a existência de ciclos de comprimento pequeno de forma a melhorar o desempenho do algoritmo de decodificação, conhecido como Algoritmo Soma-produto (SPA). Prova-se que o algoritmo SPA tem desempenho ótimo quando aplicado a grafos sem ciclos [15], [16], [17], [18]. Na presença de ciclos, a eficiência do código diminui, sendo esta inferior para um *girth* baixo. Por outro lado, é provado também que grafos de Tanner sem ciclos não geram bons códigos [19]. É necessário obedecer algumas regras na construção dos códigos LDPC para que estes possuam boas propriedades (distância mínima e *girth* elevados).

Qualquer ciclo de um grafo de Tanner tem obrigatoriamente um comprimento par, e o seu valor mínimo é 4, correspondendo a uma matriz de verificação de paridade **H** em que existem duas colunas com dois 1's na mesma posição (Figura 4.3). Uma das regras de construção da matriz **H**, para evitar a existência de ciclos de dimensão 4, consiste em garantir que quaisquer duas colunas da matriz possuam no máximo um 1 em comum.

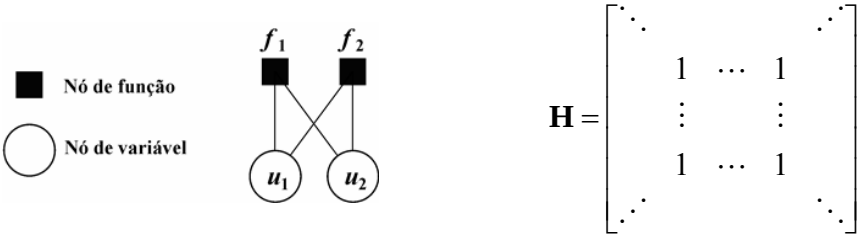


Figura 4.3: Um ciclo de comprimento 4 acompanhado de sua matriz de verificação de paridade.

Os ciclos de comprimento 6 são mais difíceis de serem identificados na matriz **H**. Na Figura 4.4 apresentamos um exemplo.

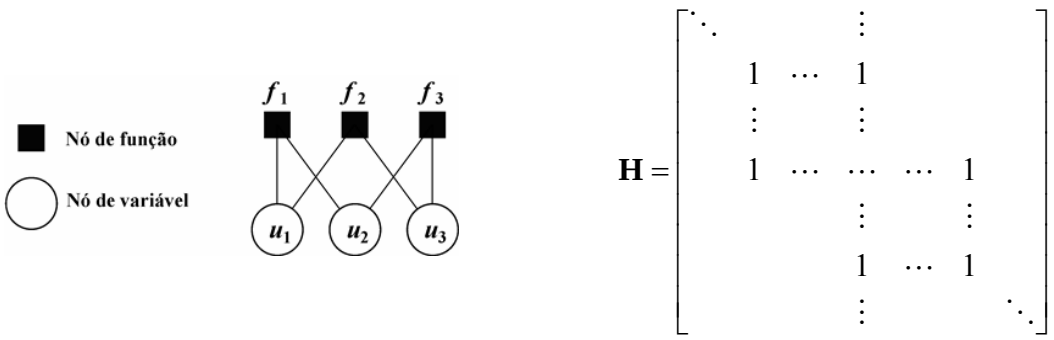


Figura 4.4: Um ciclo de comprimento 6 acompanhado de sua matriz de verificação de paridade.

Mao e Banihashemi [20] apresentam uma forma simples de determinar o *girth* associado a cada nó de variável, ou seja, o ciclo mais curto que passa por cada nó de variável. O método consiste em construir para cada nó de variável uma árvore a partir do grafo de Tanner ou da matriz **H** do código. Assim, considera-se como raiz da árvore o nó de variável cujo *girth* pretendemos determinar. A árvore é então construída, sendo que no nível *k* de descendência são incluídos todos os nós a uma distância *k* do nó raiz. O procedimento é repetido até o nível de descendência n-1, em que é incluído um nó que se encontra ligado no grafo do código a pelo menos dois nós já incluídos no nível de descendência anterior. Isto identifica a formação do primeiro ciclo, sendo $2n$ o *girth* do nó raiz.

Portanto, o método consiste em adicionar como descendentes, a cada nó *j* da árvore, todos os nós que a ele se encontram ligados no grafo de Tanner, com exceção do nó pai, terminando o algoritmo quando é encontrado um nó que é descendente de mais do que um nó diferente.

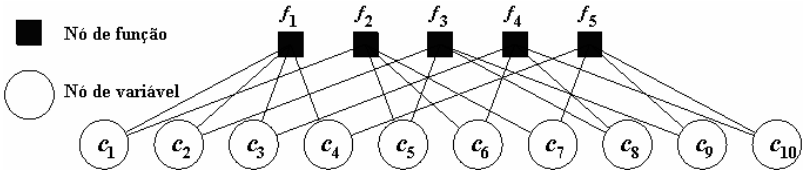


Figura 4.5: Grafo do código LDPC regular (10,5,2).

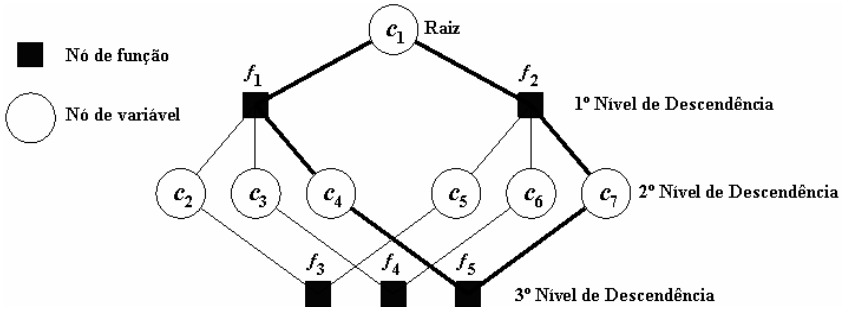


Figura 4.6: Determinação do *girth*.

Na Figura 4.6 podemos observar o método descrito aplicado ao código representado pelo seu grafo de Tanner na Figura 4.5. Podemos observar que somente no 3º nível de descendência encontra-se um nó que é descendente simultaneamente de mais de um nó, e dessa forma concluímos que o *girth* do nó c_1 é 6, que pode ser observado na Figura 4.6 pelo caminho de arestas em negrito.

McGowan e Williamson [21] apresentam um método algébrico de determinação do *girth* de cada nó de variável, baseado no conceito de matriz adjacente. Dado um código LDPC descrito por uma matriz de verificação de paridade, eles definem a sua matriz adjacente completa como:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{H} \\ \mathbf{H}^{\mathbf{T}} & \mathbf{0} \end{bmatrix}.$$

Essa definição é um pouco diferente daquela apresentada anteriormente neste trabalho, onde consideramos a matriz adjacente equivalente à matriz **H**, mas é de fácil compreensão se considerarmos que neste caso estão sendo listados todos os nós nas linhas, e todos os nós também nas colunas, obtendo uma matriz de adjacência quadrada **A**.

McGowan e Williamson provaram que o elemento (i, j) da matriz $\mathbf{A^n}$ é o número de percursos com comprimento *n* entre os nós *i* e *j* do grafo do código. Assim, cada elemento da diagonal da matriz $\mathbf{A^n}$, a_{ii}^n, representa o número de ciclos de dimensão *n* que contém o nó *i*. O método descrito permite determinar rapidamente o *girth* de cada nó de variável, bem como o *girth* do código.

4.2 CONSTRUÇÃO DE CÓDIGOS LDPC

O projeto de um código LDPC (n,k) consiste na construção da matriz de verificação de paridade **H** que atinja os objetivos pretendidos para o código, seja ele um código LDPC regular ou irregular. Existem diversos métodos para se obter esta matriz de verificação de paridade e fazendo-se algumas restrições nesta matriz, como número de 1's por coluna, taxa do código, etc., podemos criar várias famílias de códigos LDPC.

Como um exemplo dessas restrições, vamos apresentar aqui neste trabalho o conjunto de regras utilizadas por Mackay e Neal [9]. As regras são apresentadas na ordem que os autores acreditam maximizar a probabilidade de o código obtido possuir um melhor desempenho. Mas eles próprios admitem não possuir nenhuma prova disso.

As regras apresentadas por Mackay e Neal para a construção de códigos LDPC são as seguintes:

i. A matriz **H** é gerada partindo de uma matriz de zeros de dimensões $(n-k)\times n$ e aleatoriamente negando w_c bits em cada coluna (o código assim gerado poderá ser irregular);

ii. A matriz **H** é gerada criando aleatoriamente colunas de peso de Hamming w_c;

iii. A matriz **H** é gerada criando aleatoriamente colunas de peso de Hamming w_c e procurando uniformizar ao máximo o peso de Hamming w_r de cada linha;

iv. A matriz **H** é gerada com colunas de peso de Hamming w_c, linhas de peso de Hamming w_r, e não possuindo quaisquer duas colunas com mais de um 1 em comum;

v. A matriz **H** é gerada de acordo com o procedimento anterior, mas tendo como objetivo a maximização do *girth* do código;

vi. A matriz **H** é gerada de acordo com o procedimento referido em (iv.) procurando obter uma matriz **H** de característica máxima, de preferência na forma $\mathbf{H} = [\mathbf{H_1} \mid \mathbf{H_2}]$ com $\mathbf{H_1}$ ou $\mathbf{H_2}$ inversível.

Existem diversas formas de se obter uma matriz **H** eficiente para um bom código LDPC. Entretanto, existe um problema que é pouco tratado na literatura: sua codificação. Poderíamos aqui estudar diversas formas de se obter uma matriz **H** que resulta em um código excelente. Entretanto, a codificação não se resume a se obter a melhor matriz, mas sim de se obter um codificador simples, um código eficiente e um decodificador simples e com bom desempenho.

Sendo os códigos LDPC códigos de bloco lineares, podemos obter uma matriz geradora como foi feito na Seção 2.3.1 deste trabalho ($\mathbf{v} = \mathbf{uG}$). Entretanto, os métodos utilizados na construção de códigos LDPC firmam-se na obtenção da sua matriz de verificação de paridade **H**. Na maioria das vezes, a matriz **H** obtida não é sistemática nem de posto completo. A matriz **H** pode ser colocada na forma sistemática utilizando o método de Gauss, tornando a obtenção da matriz **G** muito simples e direta. Embora o problema pareça estar resolvido, na maior parte das vezes o processo de tornar a matriz **H** sistemática exige trocas de colunas, fazendo com que o código LDPC obtido seja diferente do original, mas com o mesmo desempenho.

O método de tornar a matriz **H** sistemática é de fácil execução, mas extremamente dispendioso em termos do número de operações a se realizar na codificação de cada palavra-código. Isso se deve ao fato de o processo de sistematizar a matriz **H,** na maior parte das vezes, não resultar numa matriz **G** com baixa densidade de 1's.

4.3 CODIFICAÇÃO

Como já comentado antes, a partir de qualquer matriz $\mathbf{H}$ que represente um código LDPC é possível criar uma matriz $\mathbf{G}$, matriz geradora de palavras-código. Este processo de criar palavra-código recebe o nome de codificação, em que a informação a ser transmitida é dividida em blocos de bits, com comprimento definido, e através da formula $\mathbf{v = uG}$ obtemos as palavras-código $\mathbf{v}$ referentes aos blocos de informação $\mathbf{u}$.

O processo descrito acima é utilizado em códigos de bloco lineares em geral, mas no caso dos códigos LDPC, que são códigos de bloco lineares com matrizes de verificação de paridade enormes, esse processo até pode ser utilizado, mas envolve um número muito grande de operações e a utilização de bastante memória, que torna o processo custoso, inviabilizando sua utilização. Em particular, enquanto a complexidade de decodificação via grafos de um código LDPC de comprimento n é linear em n, a complexidade de codificação é da ordem de n^2.

Existem muitos métodos alternativos para se realizar a codificação sem o uso da matriz geradora, mas a grande maioria deles envolve também um grande número de operações. O método utilizado neste trabalho é baseado nos códigos RA, *Repeat-Accumulate Codes* [12]. O codificador utilizado faz uso das propriedades do grafo para gerar as palavras-código, possibilitando uma codificação bastante simples e eficiente, com complexidade linear em n.

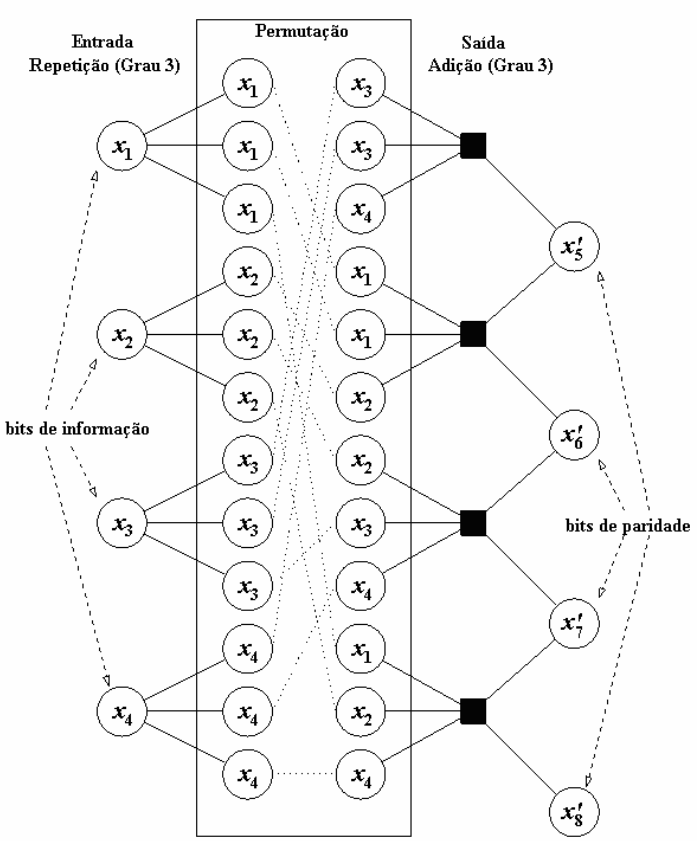


Figura 4.7: Codificador RA gerando código sistemático com taxa 4/8.

O modelo de codificador apresentado na Figura 4.7 é bastante simples. Os nós x_1, x_2, x_3 e x_4 correspondem aos bits de informação. Essa informação é triplicada gerando um novo "falso" vetor de informação, que sofre uma permutação aleatória. O vetor resultante desta permutação é somado nos nós de função, assim gerando os bits de paridade do código. Como se trata de um código de bloco linear, o código RA também pode ser representado na forma tradicional por um grafo bipartido, como mostrado na Figura 4.8.

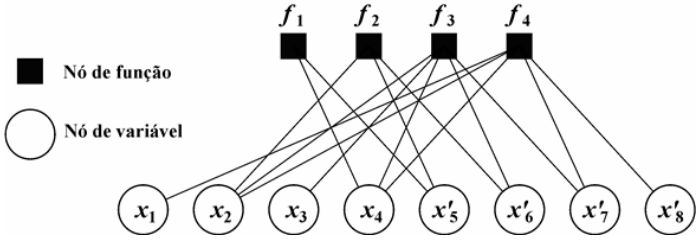


Figura 4.8: Grafo bipartido do código sistemático com taxa 4/8.

A matriz de verificação de paridade desse código é mostrada a seguir para fins didáticos.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Observando com atenção o grafo da Figura 4.8, pode-se notar que alguns bits de informação se ligam apenas a um nó de função. Isso acontece devido à permutação não ter sido muito eficiente, sendo possível através de outras permutações encontrar um código com melhor desempenho.

Foi escolhido esse tipo de codificador devido a sua simplicidade, tanto com respeito à geração do código como no que tange à obtenção da matriz **H**. É importante notar que quando a permutação resultar na situação em que um mesmo nó de variável se ligue repetidamente a um mesmo nó de função, deve-se verificar o número de vezes que esse nó de variável se liga a esse nó de função. Se esse número for par, devido à soma módulo 2 que se forma no nó de função, considera-se que esse nó de variável não contribui para esse nó de função. Veja, por exemplo, o nó de variável x_3 na Figura 4.7, que se liga duas vezes ao primeiro nó de função (de cima pra baixo). Na Figura 4.8 o nó de função f_1 representa este nó, e não existe conexão com o nó de variável x_3. O mesmo pode ser observado na matriz de verificação de paridade **H** onde a primeira linha representa todas as conexões entre o nó de função f_1 e os nós de variável. A conexão com o nó x_3 é representada pelo terceiro elemento dessa linha, e esse elemento na matriz é igual a 0, que significa a ausência de conexão.

Esta arquitetura do codificador possibilita a escolha da taxa desejada com facilidade, permitindo ainda o uso de várias taxas num mesmo sistema, criando um algoritmo que altere o grau da repetição na entrada, o grau da soma na saída, ou até mesmo os dois, fazendo o codificador variável conforme a taxa desejada.

4.4 DECODIFICAÇÃO ITERATIVA

Devido ao seu excelente desempenho, os códigos LDPC são atualmente reconhecidos como a classe de códigos que melhor se aproxima do limite de Shannon [1]. O sucesso dos códigos LDPC deve-se em grande parte ao algoritmo de decodificação iterativo, introduzido por Gallager [5] e depois redescoberto por MacKay e Neal [6], [9], que evidenciaram a sua importância.

O algoritmo apresentado por Gallager [5] tem por base a representação gráfica dos códigos LDPC, e é conhecido como Algoritmo Soma-Produto (SPA). No entanto, o seu campo de aplicação vai muito além da decodificação de códigos LDPC, abrangendo áreas do processamento de sinal, das comunicações digitais e da inteligência artificial, onde é conhecido por *Belief Propagation* devido a sua aplicação em redes Bayesianas.

A designação do algoritmo SPA por *belief propagation* é comum no contexto da decodificação de códigos LDPC. Ambos os termos são encontrados em diversos trabalhos e têm o mesmo significado.

Quando falamos em decodificação de um código, existem duas abordagens possíveis:

1º) A decodificação abrupta (*hard decoding*) considera que o conjunto de símbolos na entrada do decodificador é finito, ou seja, no caso de um código binário na entrada do decodificador teremos apenas bits, a decisão se cada bit recebido foi 0 ou 1 sendo tomada antes da decodificação. Depois é escolhida a palavra-código mais próxima do vetor recebido;

2º) A decodificação suave (*soft decoding*) considera a distribuição probabilística dos símbolos recebidos, sem a decisão prévia bit a bit, e toma sua decisão baseada na palavra inteira.

Os algoritmos do tipo *hard decoding* são na sua grande maioria menos complexos e menos exigentes computacionalmente. Também é verdade que levando-se em conta a distribuição probabilística dos dados recebidos pelo decodificador, os algoritmos *soft decoding* apresentem um melhor desempenho em termos de uma menor taxa de erros (BER). Por isso, neste trabalho, optamos por utilizar um decodificador do tipo *soft decoding*, e o algoritmo utilizado neste trabalho foi o SPA.

4.4.1 ALGORITMO SPA

O algoritmo SPA opera sobre o grafo de Tanner de um código de bloco binário linear, e funciona, basicamente, como um algoritmo de passagem de mensagens entre nós. De uma forma simplista, cada nó pode ser visto como um processador de mensagens recebidas dos seus vizinhos (nós ligados a ele), aos quais devolve posteriormente mensagens atualizadas. As mensagens recebidas por um nó de variável, dos nós de função que a ele se encontram ligados, ou as enviadas desse nó de variável para esses nós de função, não são mais do que “opiniões” sobre o valor lógico desse nó de variável. Essas “opiniões” são

expressas em termos da distribuição probabilística dos símbolos recebidos pelo decodificador.

O algoritmo SPA apresenta na decodificação de códigos LDPC o melhor desempenho, tendo Chung, Forney, Richardson e Urbanke [22] demonstrado ser possível se aproximar (a menos de 0.0045dB) da capacidade de um canal AWGN considerando, no limite, um código de comprimento infinito. Para a decodificação *soft decoding* de códigos LDPC, o algoritmo SPA é descrito no domínio linear, que é a sua forma original [5], e no domínio logarítmico [4]. Este último apresenta vantagens na implementação, mas possui desempenho um pouco inferior. Nesta dissertação, consideramos apenas a implementação do SPA no domínio linear.

Consideremos um código binário LDPC (n,k) onde as palavras-código são transmitidas através de um canal AWGN. Os bits da palavra-código $v_i \in \{0,1\}$, com $i = \{1,2,\ ...,n\text{-}1,n\}$, são modulados em BPSK, fazendo-lhes corresponder os símbolos $x_i = (-1)^{v_i+1}$. Os símbolos recebidos ($\mathrm{y_i}$) têm somados a eles o ruído Gaussiano (η_i com média nula e variância σ^2), portanto temos que $y_i = x_i + \eta_i$.

Como já foi dito, o algoritmo SPA baseia-se na passagem de informação entre nós. Cada nó de variável envia para cada um dos nós de função a ele ligados toda a informação que ele recebeu com exceção da informação que o nó de função já possui. Ou seja, sendo f_i um nó de função ligado ao nó de variável x_i, a informação enviada por este nó de variável para o nó de função inclui a informação que recebeu do canal, y_i, e a informação recebida de todos os outros nós de função ligados a ele, designada por **informação extrínseca**.

Da mesma forma, cada nó de função envia para cada nó de variável ao qual se encontra ligado toda a informação extrínseca que ele recebeu dos restantes nós de variável a ele ligados. No entanto, neste caso, a informação enviada tem que verificar a restrição associada a esse nó de função.

Exemplo 4.2: Na Figura 4.9, à esquerda, a mensagem que o nó de variável x_1 envia para o nó de função f_1 é formada pela informação recebida do canal e pela informação extrínseca recebida dos nós de função f_2 e f_3 na etapa anterior.

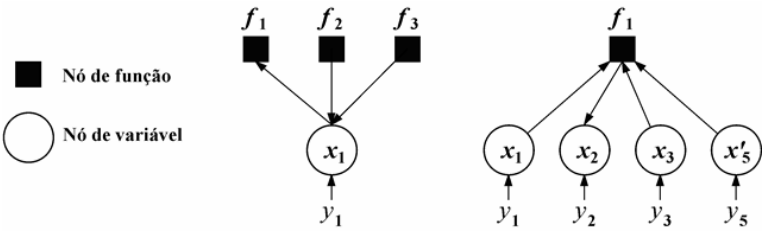


Figura 4.9: Grafo parcial do grafo correspondente ao código de Hamming de taxa 4/8.

Da mesma forma, na Figura 4.9, à direita, a informação que o nó de função f_1 envia para o nó de variável x_2 é formada pela informação extrínseca recebida dos nós x_1, x_3 e x'_5 na etapa anterior.

Tratando-se de um algoritmo do tipo *soft decoding*, que considera a distribuição probabilística dos símbolos recebidos, facilmente se conclui que as mensagens a serem transmitidas entre os nós são nada mais do que probabilidades. O objetivo do algoritmo SPA é maximizar a probabilidade *a posteriori* de um bit v_i, da palavra decodificada $\hat{\mathbf{v}}$, ser 1 ou 0 tendo em conta o vetor recebido $\mathbf{y}$ e o conjunto de restrições S_i que o bit v_i tem que satisfazer. Portanto, pretendemos calcular:

$$P\left(v_i = 0 \mid \mathbf{y}, S_i\right), \tag{4.1}$$

e

$$P\left(v_i = 1 \mid \mathbf{y}, S_i\right). \tag{4.2}$$

Designamos por:

- $P_i = P\left(v_i = 1 \mid y_i\right)$ com y_i o *i*-ésimo símbolo do vetor recebido $\mathbf{y}$;
- $v_{tr} \rightarrow$ o bit da palavra-código correspondente ao *t*-ésimo termo da *r*-ésima equação de verificação de paridade (Por exemplo, a partir da Equação 2.8 e com a matriz **H** da Seção 4.3, a primeira (*r* = 1) equação de verificação de paridade é: $v_4 + v_5 = 0$. Assim, a partir das restrições mencionadas acima, apenas os termos $v_{11} = v_4$ e $v_{12} = v_5$);
- $w_r(r) \rightarrow$ o peso de Hamming da *r*-ésima linha de **H**;
- $w_c(t) \rightarrow$ o peso de Hamming da *t*-ésima coluna de **H**;
- $y_{tr} = (-1)^{v_{tr}+1} + \eta$, com η AWGN → Símbolo recebido correspondente a v_{tr};

- $P_{tr} = P(v_{tr} = 1 \mid y_{tr})$.

Assumindo que os símbolos do vetor recebido $\mathbf{y}$ são estatisticamente independentes, Gallager [5], [8], demonstrou um teorema que estabelece que a razão entre as probabilidades *a posteriori* (4.1) e (4.2) é dada por:

$$\frac{P(v_i = 0 \mid \mathbf{y}, S_i)}{P(v_i = 1 \mid \mathbf{y}, S_i)} = \frac{1 - P_i}{P_i} \prod_{r=1}^{w_c(r)} \left[\frac{1 + \prod_{\substack{t=1 \\ t \neq i}}^{w_r(r)} (1 - 2P_{tr})}{1 - \prod_{\substack{t=1 \\ t \neq i}}^{w_r(r)} (1 - 2P_{tr})} \right]. \tag{4.3}$$

Para provar esse teorema, usou-se a probabilidade de se ter um número par de 1's em um determinado nó de função, que é dada por:

$$\frac{1 + \prod_{t=1}^{n} (1 - 2P_t)}{2}. \tag{4.4}$$

Considerando $v_i = 0$, os bits iguais a 1 da palavra-código que fazem parte da equação de verificação de paridade envolvendo v_i deverão ser em número par, a fim de que a equação seja satisfeita. Assim, para uma equação de verificação de paridade *r* que depende de v_i, se $v_i = 0$, então a probabilidade de existir um número par de 1's entre os bits envolvidos nessa equação de verificação de paridade é:

$$\frac{1}{2} + \frac{1}{2} \prod_{\substack{t=1 \\ t \neq i}}^{w_r(r)} (1 - 2P_{tr}). \tag{4.5}$$

Logo, a probabilidade de todas as equações de verificação de paridade dependentes de v_i serem satisfeitas para $v_i = 0$ é:

$$P(v_i = 0 \mid \mathbf{y}, S_i) = \prod_{r=1}^{w_c(r)} \left[\frac{1}{2} + \frac{1}{2} \prod_{\substack{t=1 \\ t \neq i}}^{w_r(r)} (1 - 2P_{tr}) \right]. \tag{4.6}$$

De forma idêntica, podemos concluir que para $v_i = 1$,

$$P(v_i = 1 \mid \mathbf{y}, S_i) = \prod_{r=1}^{w_c(r)} \left[\frac{1}{2} - \frac{1}{2} \prod_{\substack{t=1 \\ t \neq i}}^{w_r(r)} (1 - 2P_{tr}) \right]. \tag{4.7}$$

À semelhança de Gallager, facilmente podemos concluir que o cálculo da razão (Equação 4.3) entre as probabilidades *a posteriori* é demasiadamente complexo. Gallager

propôs um algoritmo iterativo para determinar essas probabilidades *a posteriori*, isto é, o algoritmo SPA.

Com vista a descrever o algoritmo é necessária alguma notação adicional. Assim, designemos:

- $q_{ij}(b)$, onde $b \in \{0,1\}$ → mensagem que é enviada do nó de variável *i* para o nó de função *j*, indicando a probabilidade de $v_i = b$, baseada na informação recebida do canal, y_i, e em todos os nós de função dependentes de v_i, com exceção do nó de função $j = i$;
- $r_{ji}(b)$, onde $b \in \{0,1\}$ → mensagem que é enviada do nó de função *j* para o nó de variável *i*, indicando a probabilidade de $v_i = b$, baseada em todos os nós de variável ligados ao nó de função j, com exceção do nó $i = j$;
- $Q_i(b)$, onde $b \in \{0,1\}$ → pseudo-probabilidade *a posteriori* de $v_i = b$.

Tendo em conta a definição de $q_{ij}(b)$, $r_{ji}(b)$ e $Q_i(b)$, podemos então concluir que:

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{\substack{t=1 \\ t\neq i}}^{w_r(j)}\left(1-2P_{tj}\right) = \frac{1}{2} + \frac{1}{2}\prod_{\substack{t=1 \\ t\neq i}}^{w_r(j)}\left(1-2q_{tj}(1)\right), \tag{4.8}$$

$$r_{ji}(1) = \frac{1}{2} - \frac{1}{2}\prod_{\substack{t=1 \\ t\neq i}}^{w_r(j)}\left(1-2P_{tj}\right) = \frac{1}{2} - \frac{1}{2}\prod_{\substack{t=1 \\ t\neq i}}^{w_r(j)}\left(1-2q_{tj}(0)\right), \tag{4.9}$$

e podemos expressar a razão (4.3) como:

$$\frac{P(v_i = 0 \mid \mathbf{y}, S_i)}{P(v_i = 1 \mid \mathbf{y}, S_i)} = \frac{1-P_i}{P_i}\prod_{\substack{j=1 \\ j\neq i}}^{w_c(i)}\left[\frac{r_{ji}(0)}{r_{ji}(1)}\right]. \tag{4.10}$$

A partir da definição de $q_{ij}(b)$, temos:

$$q_{ij}(0) = (1-P_i)\prod_{\substack{r=1 \\ r\neq j}}^{w_c(i)}\left(r_{ri}(0)\right), \tag{4.11}$$

$$q_{ij}(1) = P_i\prod_{\substack{r=1 \\ r\neq j}}^{w_c(i)}\left(r_{ri}(1)\right). \tag{4.12}$$

A decisão sobre o valor binário do bit *i* da mensagem recebida é então tomada tendo por base as pseudo-probabilidades *a posteriori*,

$$Q_i(0) = (1 - P_i) \prod_{j=1}^{w_c(i)} \left(r_{ji}(0) \right), \tag{4.13}$$

$$Q_i(1) = P_i \prod_{j=1}^{w_c(i)} \left(r_{ji}(1) \right), \tag{4.14}$$

onde o bit *i* da palavra decodificada $\hat{\mathbf{v}}$ é dado por:

$$v_i \begin{cases} = 1, & \text{para} \quad Q_i(1) > 0,5 \\ = 0, & \text{para} \quad Q_i(1) < 0,5 \end{cases}. \tag{4.15}$$

As Equações 4.11 e 4.12, bem como 4.13 e 4.14, deverão ser normalizadas da forma a seguir: $q_{ij}(1) + q_{ij}(0) = 1$ e $Q_i(1) + Q_i(0) = 1$.

O algoritmo de decodificação pode ser descrito do seguinte modo:

Os passos de 1 a 4 do algoritmo devem ser executados sempre que o nó de variável *i* e o de função *j* satisfizerem na matriz de verificação de paridade a igualdade $h_{ij} = 1$. Esta restrição, aliada à esparsidade de **H**, torna a decodificação viável computacionalmente.

1. Inicialização:

   $q_{ij}(0) = (1 - P_i)$,

   $q_{ij}(1) = P_i$.

2. Calcular a mensagem que o nó de função *j* envia para o nó de variável *i*:

   $$r_{ji}(0) = \frac{1}{2} + \frac{1}{2} \prod_{\substack{t=1 \\ t \neq i}}^{n} \left(1 - 2q_{tj}(1) \right),$$

   $r_{ji}(1) = 1 - r_{ji}(0)$.

3. Calcular a mensagem que o nó de variável *i* envia para o nó de função *j*:

   $$q_{ij}(0) = k_{ij} (1 - P_i) \prod_{\substack{r=1 \\ r \neq j}}^{n-k} \left(r_{ri}(0) \right),$$

   $$q_{ij}(1) = k_{ij} P_i \prod_{\substack{r=1 \\ r \neq j}}^{n-k} \left(r_{ri}(1) \right),$$

   onde as constantes k_{ij} são escolhidas de forma que $q_{ij}(1) + q_{ij}(0) = 1$.

4. Calcular as pseudo-probabilidades *a posteriori*:

   $$Q_i(0) = k_i (1 - P_i) \prod_{r=1}^{n-k} \left(r_{ri}(0) \right),$$

$$Q_i(1) = k_i P_i \prod_{r=1}^{n-k} \left(r_{ri}(1) \right),$$

onde as constantes k_i são escolhidas de forma que $Q_i(1) + Q_i(0) = 1$.

5. $\forall\, i$, fazer:

$$v_i \begin{cases} = 1, & \text{para} \quad Q_i(1) > 0,5 \\ = 0, & \text{para} \quad Q_i(1) < 0,5 \end{cases}.$$

Se a palavra decodificada verificar as equações de verificação de paridade ($\mathbf{Hv^T = 0_{(n-k)\times 1}}$) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar ao passo número 2.

Em caso de parada, devolver:

- Palavra decodificada $\hat{\mathbf{v}}$ se $\mathbf{Hv^T = 0_{(n-k)\times 1}}$;
- Erro se (número de iterações = max_iterações).

Para utilizar o algoritmo descrito, é necessário especificar o modelo de canal que será utilizado. No caso do canal AWGN, podemos determinar suas probabilidades da seguinte forma:

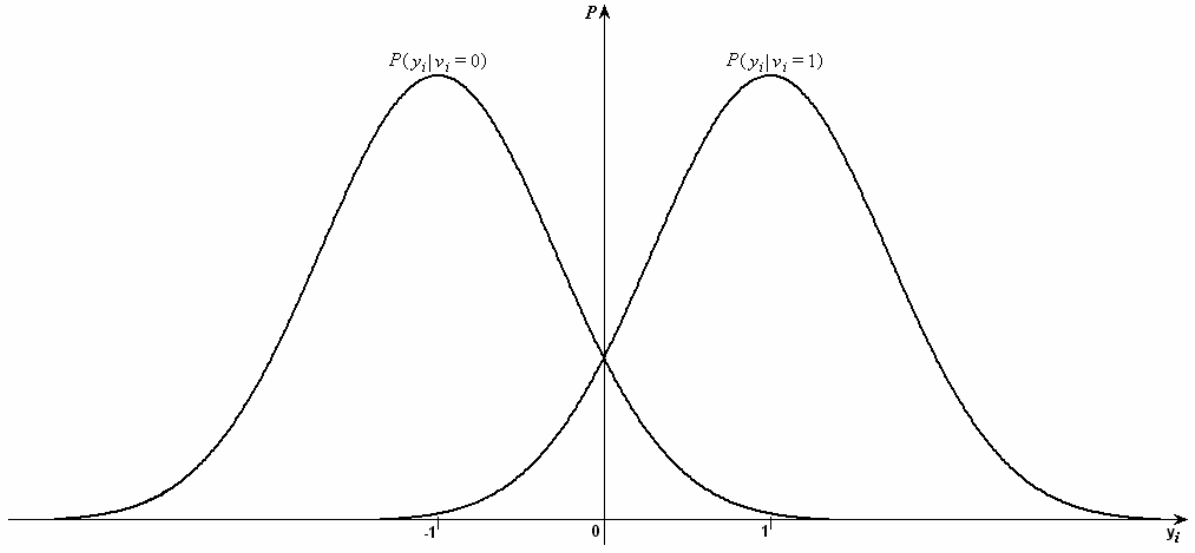


Figura 4.10: Densidades de probabilidade para o modelo de canal Gaussiano.

Na Figura 4.10 temos as densidades de probabilidades $P(y_i \mid v_i = 0)$ e $P(y_i \mid v_i = 1)$ das transmissões de 0's e 1's no canal AWGN. Pode-se observar que quando 0 é transmitido pelo canal o valor a ser recebido é uma variável aleatória Gaussiana com média -1 e variância dependente da intensidade do ruído, e no caso de ser 1 o bit transmitido, tem-se

uma variável aleatória Gaussiana com média +1 e variância também dependente da intensidade do ruído. Portanto temos:

$$P_i = P(v_i = 1 \mid y_i) = \frac{P(y_i \mid v_i = 1)P(v_i = 1)}{P(y_i)}, \tag{4.16}$$

$$P(y_i) = P(y_i \mid v_i = 0)P(v_i = 0) + P(y_i \mid v_i = 1)P(v_i = 1). \tag{4.17}$$

Substituindo a Equação 4.17 em 4.16, temos:

$$P_i = \frac{P(y_i \mid v_i = 1)P(v_i = 1)}{P(y_i \mid v_i = 0)P(v_i = 0) + P(y_i \mid v_i = 1)P(v_i = 1)},$$

onde $$P(v_i = 0) = P(v_i = 1) = \frac{1}{2}.$$

Fazendo as devidas simplificações temos:

$$P_i = \frac{P(y_i \mid v_i = 1)}{P(y_i \mid v_i = 0) + P(y_i \mid v_i = 1)},\ P_i = \frac{1}{1 + \frac{P(y_i \mid v_i = 0)}{P(y_i \mid v_i = 1)}}. \tag{4.18}$$

Sabendo que as Gaussianas são descritas por:

$$P(y_i \mid v_i = 1) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i - 1)^2}{2\sigma^2}},\ P(y_i \mid v_i = 0) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i + 1)^2}{2\sigma^2}}, \tag{4.19}$$

temos que:

$$\mathrm{P_i} = \frac{1}{1 + \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i+1)^2}{2\sigma^2}} \frac{\sigma\sqrt{2\pi}}{1} e^{\frac{(y_i-1)^2}{2\sigma^2}}} = \frac{1}{1 + e^{\frac{-(y_i+1)^2 + (y_i-1)^2}{2\sigma^2}}} = \frac{1}{1 + e^{\frac{-y_i^2 - 2y_i - 1 + y_i^2 - 2y_i + 1}{2\sigma^2}}}.$$

Finalmente concluímos que:

$$P_i = \frac{1}{1 + e^{-\frac{2y_i}{\sigma^2}}}. \tag{4.20}$$

Exemplo 4.3: Utilizando o algoritmo descrito e as probabilidades calculadas com base no canal Gaussiano, testamos o desempenho do algoritmo SPA frente ao decodificador de máxima verossimilhança (MLD) apresentado no Capitulo 2 deste trabalho.

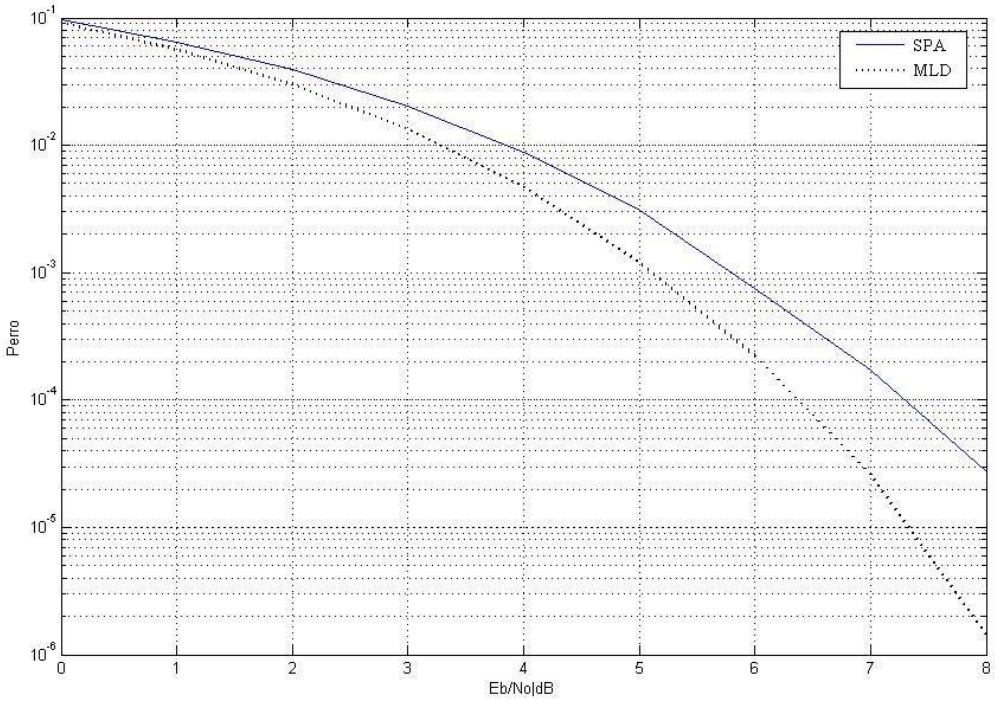


Figura 4.11: Gráfico comparativo entre o desempenho do decodificador SPA e o decodificador de máxima verossimilhança apresentado no Capitulo 2, ambos para o código de Hamming taxa 4/8.

Como o exemplo utilizado na Figura 4.11 para ilustrar o funcionamento do algoritmo SPA foi um código linear de bloco com palavras com comprimento de 8 bits, era esperado um desempenho inferior do decodificador SPA. O código é muito "pequeno" e possui *girth* igual a 4, e isso faz com que o algoritmo apresente baixo desempenho.

4.5 CONCLUSÕES DO CAPÍTULO

Neste capítulo apresentamos os códigos LDPC, suas classificações, construção, codificação e decodificação. No próximo capítulo será apresentada a proposta de utilização de códigos LDPC em canais de gravação magnética. Baseado no canal de resposta parcial "dicode", será construído um código, um codificador e um decodificador específicos para o esse canal. Também será apresentado o resultado de simulação mostrando a eficiência da proposta.

CAPÍTULO 5

CÓDIGOS LDPC E CANAL DE GRAVAÇÃO MAGNÉTICA

Neste capítulo, vamos falar sobre a proposta desta dissertação de utilizar códigos LDPC em gravações em meios magnéticos. Vamos apresentar as características do canal de gravação magnética, e descrever as modificações no codificador e decodificador do código LDPC.

Consideramos a cabeça de gravação magnética, o meio de gravação magnético, e a cabeça de leitura magnética como um único sistema linear. Um modelo bastante aceito como a resposta ao degrau deste canal é conhecido como pulso de Lorentz, e pode ser descrito de uma forma simplificada pela equação:

$$h(t) = \frac{1}{1+\left(\frac{2t}{PW_{50}}\right)^2}, \tag{5.1}$$

onde PW_{50} é a largura do pulso na metade da sua amplitude.

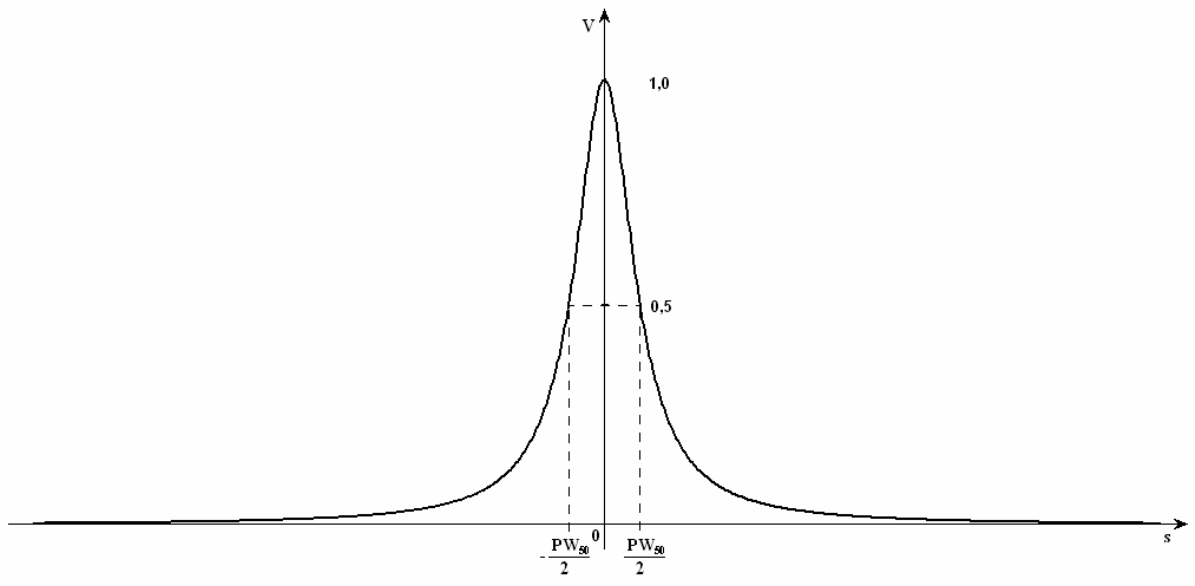


Figura 5.1: Pulso de Lorentz.

Para realizarmos a gravação de dados binários nesse canal magnético, temos que utilizar algum tipo de modulação. Vamos recorrer à modulação polar **NRZ** (pulso polarizado sem retorno a zero), onde o dígito 1 é representado pelo sinal positivo (+1) e o dígito 0 pelo sinal negativo (-1). Aqui utilizamos amplitude unitária para uma maior simplicidade nas equações.

Podemos escrever a corrente elétrica, $i(t)$, responsável pela gravação no canal como:

$$i(t) = \sum_{k=0}^{\infty} (2b_k - 1)\, x(t - kT), \tag{5.2}$$

onde b_k corresponde à seqüência binária (0's e 1's) a ser gravada no canal, e $x(t)$ é o pulso retangular definido como:

$$x(t) = \begin{cases} 1, & 0 \le t < T \\ 0, & \text{qualquer outro } t \end{cases}.$$

5.1 MODELO DE CANAL 1 - D

Antes de descrever mais profundamente os canais de resposta parcial [23], [24], temos que relembrar o princípio da superposição. Ou seja, quando uma transição positiva no instante de tempo t resulta em um pulso $h(t)$, e uma transição negativa no instante de tempo $t + \Delta T$ resulta em um pulso $-h(t + \Delta T)$, então a resposta de uma transição positiva seguida por uma negativa será $h(t) - h(t + \Delta T)$. Para evitar interferência intersimbólica, deve existir uma separação mínima entre duas transições consecutivas. Essa separação mínima depende da cabeça de gravação, do tipo de mídia utilizado e da cabeça de leitura magnética [25], [26].

Agora vamos considerar que a informação seja dada por $i(t)$. Portanto, temos das Equações 5.1 e 5.2 que a tensão elétrica na cabeça de leitura é dada por:

$$r(t) = h(t) * \frac{d}{dt}\, i(t), \tag{5.3}$$

$$r(t) = h(t) * \frac{d}{dt} \sum_{k=0}^{\infty} (2b_k - 1)\, x(t - kT), \tag{5.4}$$

seja $s(t)$ a função degrau unitário. Então o pulso retangular com largura T é $x(t) = s(t) - s(t - T)$, e a Equação 5.4 pode ser reescrita em termos de $s(t)$ como:

$$r(t) = h(t) * \frac{d}{dt} \sum_{k=0}^{\infty} (2b_k - 1)\left[s(t - kT) - s(t - (k+1)T) \right]$$

$$= h(t) * \sum_{k=0}^{\infty} (2b_k - 1)\left[\delta(t - kT) - \delta(t - (k+1)T) \right]$$

$$= \sum_{k=0}^{\infty} (2b_0 - 1)\left[h(t - kT) - h(t - (k+1)T) \right]$$

$$= \sum_{k=0}^{\infty} (2b_k - 1) h(t - kT) - \sum_{k'=1}^{\infty} (2b_{k'-1} - 1) h(t - k'T)$$

$$= (2b_0 - 1) h(t) + \sum_{k=1}^{\infty} (2b_k - 2b_{k-1}) h(t - kT).$$

Ou seja,
$$r(t) = 2\sum_{k=0}^{\infty} c_k h(t - kT), \tag{5.5}$$

onde
$$c_k = \begin{cases} b_k - b_{k-1}, & \text{para } k \geq 1 \\ b_0 - 1/2, & \text{para } k = 0 \end{cases}. \tag{5.6}$$

Se $r(t)$ é amostrado em intervalos múltiplos de *T*, então:

$$r(nT) = 2\sum_{k=0}^{\infty} c_k h((n-k)T). \tag{5.7}$$

Agora a duração *T* deve ser escolhida de forma que nos instantes de amostragem tenhamos

$$h([n-k]T) = \begin{cases} 1, & \text{para } n = k \\ 0, & \text{para } n \neq k \end{cases}. \tag{5.8}$$

Substituindo a Equação 5.8 em 5.7, temos:

$$r(nT) = 2c_n. \tag{5.9}$$

Através das Equações 5.6 e 5.9, podemos concluir que este canal magnético pode ser visto como um canal de resposta parcial, cuja função de transferência é dada pelo polinômio $F(D) = 1 - D$.

Devemos observar que, para a entrada bipolar (±1), as possíveis amplitudes na saída do canal $1 - D$ são: 0, +2 e –2. Por outro lado, esse canal apresenta uma memória característica dos canais de resposta parcial. Nesse caso, a memória é unitária, e as possíveis seqüências na saída do canal podem ser representadas por caminhos em uma treliça de dois estados [26]. A detecção da seqüência transmitida pelo canal pode ser realizada pelo algoritmo de Viterbi com base nessa treliça. A treliça associada ao canal $1 - D$ é mostrada na Figura 5.2, onde nos rótulos a/b, a representa o símbolo bipolar na entrada do canal e b a sua saída.

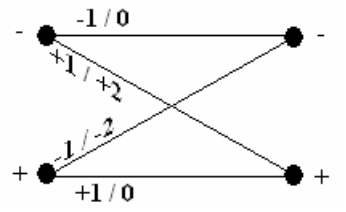


Figura 5.2: Treliça associada ao canal $1 - D$ sem pré-codificação.

Um processamento bastante utilizado em sinalização de resposta parcial é a pré-codificação da seqüência binária a ser submetida ao canal [26]. A idéia é submeter essa seqüência, antes de transmiti-la, a um filtro (pré-codificador) que inverte a função de transferência do canal de resposta parcial no domínio binário. Para o canal $1 - D$, a função de transferência do pré-codificador é: $1/1 \oplus D$, onde $\oplus$ denota soma módulo 2. O circuito envolvendo o pré-codificador e o canal $1 - D$ é mostrado na Figura 5.3.

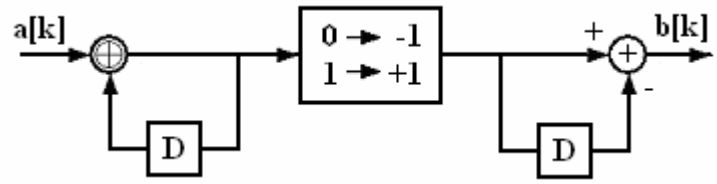


Figura 5.3: Circuito com pré-codificador e canal $1 - D$.

A seqüência binária que se forma na saída desse filtro é transformada para o formato polar NRZ e então submetida ao canal. Com isto, é estabelecida uma relação *um-para-um* entre a seqüência binária original e a seqüência que se forma na saída do canal (ternária no caso do canal $1 - D$). Com o pré-codificador, a treliça do canal $1 - D$ é aquela mostrada na Figura 5.4.

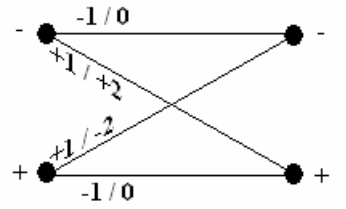


Figura 5.4: Treliça associada ao canal $1 - D$ com pré-codificação.

Devemos notar a inversão dos rótulos nos dois ramos inferiores em comparação com a Figura 5.2, em conseqüência da qual se tem a seguinte relação *um-para-um*:

$$a[k] = \frac{|b[k]|}{2} \pmod 2.$$

Com isso, a detecção da seqüência transmitida pelo canal pode ser realizada pelo método *símbolo-a-símbolo*. Essa importante vantagem obtida com a pré-codificação é explorada e

encontra-se no núcleo da proposta do código LDPC apresentada nesta dissertação para o canal $1-D$.

De um modo mais geral, canais de resposta parcial podem ser representados pelas suas funções de transferência no domínio discreto através de:

$$F(D)=\sum_{i=0}^{N} f_i D^i \ , \ f_0=1; \ f_i \ \text{ inteiro} \tag{5.10}$$

onde D é a unidade de atraso de um período *T*, correspondendo ao atraso de um símbolo. Escolhendo-se agora menores valores para a duração *T*, ou seja, aumentando-se a densidade de gravação magnética, é possível mostrar, por um procedimento semelhante ao anterior (Equações 5.3 a 5.9), que o canal de gravação magnética pode ser modelado pelos canais de resposta parcial [26]:

$$F(D)=(1-D)(1+D)^n, \ n=0,1,2,... \tag{5.11}$$

Para $n=0$, temos $F(D)=1-D$ ou o canal de resposta parcial "*dicode*". O canal para $n=1$ é denominado canal de resposta parcial classe 4 (PR4), e quando $n=2$ o canal é denominado canal de resposta parcial classe 4 estendido (EPR4) [24]. Um maior valor de *n* está associado a uma maior densidade de gravação. Nesta dissertação, consideraremos apenas o canal $1-D$.

5.2 CONSTRUÇÃO DE CÓDIGOS LDPC PARA O CANAL *1 - D*

O projeto de um código LDPC (n,k) para o canal $1-D$, semelhante ao caso do canal binário, consiste na construção da matriz de verificação de paridade **H**. O objetivo aqui é gerar uma matriz de verificação de paridade **H** que resulte num bom desempenho para o canal $1-D$**,** mantendo uma certa simplicidade nos processos de codificação e decodificação.

Na nossa proposta, visando à simplicidade, faremos, por um lado, uso da estrutura do RA que possui os bits de paridade já pré-codificados para o canal $1-D$. Por outro lado, para se ter o benefício na relação *um-para-um* apresentada anteriormente, devemos modificar os processos de codificação e decodificação da estrutura do RA, *Repeat-Accumulate Codes* [12]. Na Figura 5.5 é apresentado o diagrama de bloco do esquema proposto desde a codificação até a decodificação.

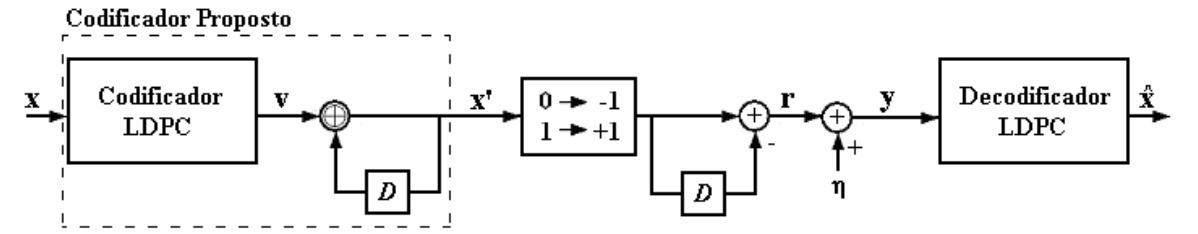


Figura 5.5: Diagrama de bloco com o codificador, canal *1 - D* e decodificador.

5.3 CODIFICAÇÃO

O processo apresentado na Figura 5.6, utilizando o codificador do tipo RA modificado, apresenta grande simplicidade porque possibilita a geração das palavras-código com um pequeno número de operações. Além disso, podemos obter a matriz de verificação de paridade $\mathbf{H}$ diretamente do codificador.

O codificador consiste em repetidores de bits, um permutador e somadores módulo 2. Cada bit do vetor de informação, que tem tamanho k, é repetido um número α de vezes, resultando em um vetor de tamanho igual a α vezes k. Esse vetor é permutado de forma aleatória, gerando um vetor com os bits de informação repetidos e "misturados". Esse novo vetor é dividido em grupos de comprimento β bits e esses são somados em um somador módulo 2 formando os $n-k$ bits de verificação de paridade. E por fim, os k bits de informação e os $n-k$ bits de paridade são submetidos a uma pré-codificação para o canal $1-D$. A pré-codificação dos bits de paridade já faz parte da proposta RA original. A pré-codificação dos bits de informação é feita através de um processo "zigzag" nesses bits, que será descrito a seguir.

Na Figura 5.6 é apresentado como exemplo ilustrativo um modelo de um codificador $(12,9)$ segundo a nova proposta, onde os nós x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7 e x_8 correspondem aos bits de informação. Neste caso esta informação é triplicada gerando um novo "falso" vetor de informação, que sofre uma permutação aleatória. O vetor resultante desta permutação é somado (modulo 2) de nove em nove bits nos nós de função, e os $n-k$ bits gerados (aqui chamados *pseudo-bits de paridade*) formam a parte final de uma seqüência (aqui chamada seqüência *pseudo-codificada*) de comprimento n que inicia com os k bits de informação. Agora temos que fazer a pré-codificação dessa seqüência para gerar as palavras-código a serem submetidas ao canal $1-D$. Como explicado anteriormente, a função da pré-codificação é obter uma relação de *um-para-um* entre a saída do canal e a seqüência que passa pela pré-codificação.

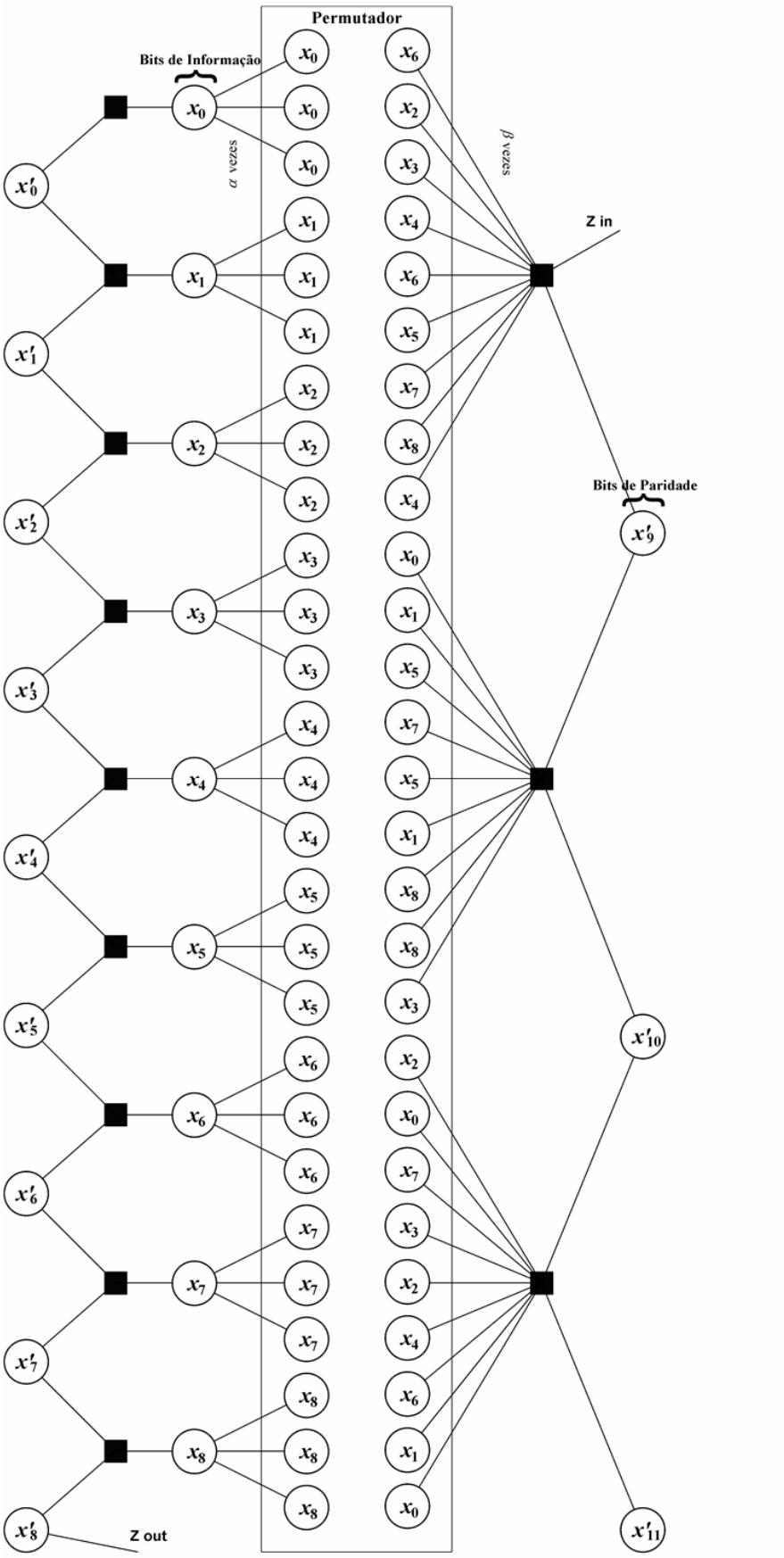


Figura 5.6: Exemplo de um codificador RA modificado para o canal *1 - D* .

Nesse codificador podemos observar o "zigzag" com os nós de variáveis x_0 até o x_8 originando novos nós de variáveis x'_0 até x'_8.Observe que o último nó de variável x'_8 se liga ao primeiro nó de função (do lado direito de cima para baixo), continuando o processo de pré-codificação gerando os bits de paridade x'_9, x'_{10} e x'_{11} da palavra-código. Uma observação importante é que a palavra-código transmitida $x'_0, x'_1, \ldots, x'_{11}$ é justamente o resultado da pré-codificação da seqüência *pseudo-codificada*. Este exemplo é apenas didático, pois o código é muito "pequeno", não tendo um bom desempenho para uma aplicação prática.

5.4 DECODIFICAÇÃO

A pré-codificação e a passagem pelo canal $1-D$ da palavra-código nos leva a utilizar uma outra matriz de verificação de paridade $\mathbf{H}$, que considera como palavra-código para decodificação a seqüência pseudo-codificada, visto que há uma relação *um-para-um* entre esta última e a seqüência recebida para decodificação. Nesses termos, para o processo de decodificação, é necessário desconsiderar o processo de "zigzag" usado na codificação. Isso torna o processo de decodificação menos custoso, porque a matriz obtida sem o "zigzag" é uma matriz com esparsidade maior. Portanto, para obtermos a matriz de verificação de paridade a ser adotada na decodificação, temos que recorrer ao esquema apresentado na Figura 5.7.

Quando analisamos as diferenças entre as duas estruturas, fica muito claro que a estrutura sem o "zigzag" fornece uma matriz de verificação de paridade mais esparsa. Isso ocorre devido aos nós de variável possuírem menor número de ligações com os nós de função. E além disso podemos verificar que agora vamos trabalhar com uma matriz análoga à apresentada no Capítulo 2, $\mathbf{H} = \left[\mathbf{P}^{\mathbf{T}} \mid \mathbf{I_k}\right]$.

Exemplo 5.1: Vamos considerar que estamos transmitindo a seqüência de bits de informação $\mathbf{x} = [1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$, utilizando o codificador apresentado na Figura 5.6. obtemos como palavra-código $\mathbf{x}' = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0]$. Submetendo ao canal $1-D$, utilizando o decodificador da Figura 5.7 e considerando a decodificação correta teremos como palavra-código $\mathbf{v}' = [1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1]$ resultando em $\hat{\mathbf{x}} = [1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$.

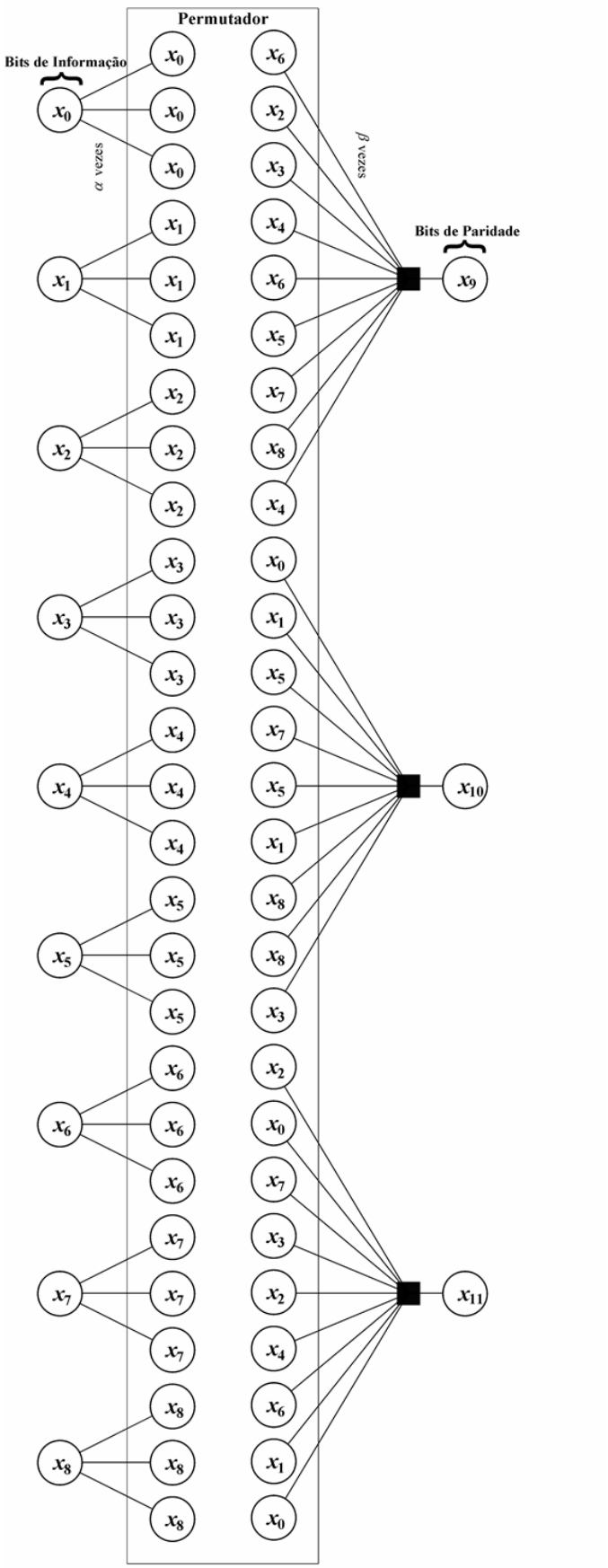


Figura 5.7: Estrutura utilizada no decodificador para o canal *1 - D* .

O algoritmo utilizado na decodificação foi o algoritmo SPA. Mas como o canal em questão não é mais o canal AWGN, foram necessárias algumas modificações referentes às probabilidades *a priori*.

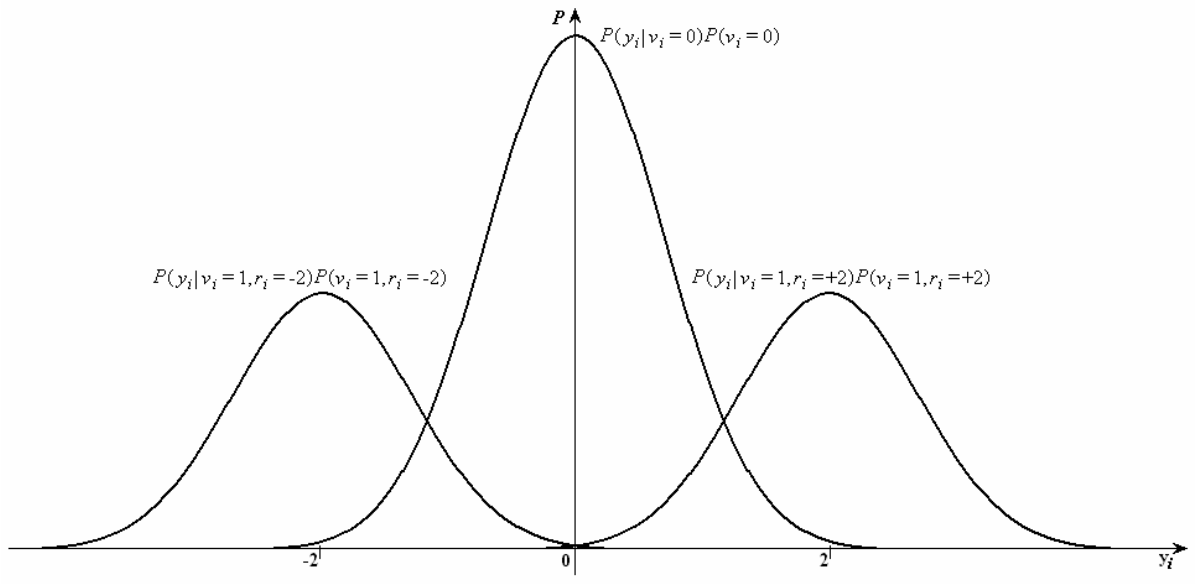


Figura 5.8: Densidades de probabilidade para o modelo de canal *1 - D*.

Diferente do canal AWGN, nesse modelo o y_i possui três valores de amplitude: -2, 0 e $+2$. Desconsiderando o ruído, quando y_i for igual a $+2$ ou -2, segundo o mapeamento *um-para-um* supramencionado, tem-se que o bit 1 foi o transmitido, e quando y_i for igual a 0 significa que o bit 0 foi transmitido. As funções de distribuição de probabilidades condicionadas e ponderadas para o canal pré-codificado, ilustradas na Figura 5.8, são:

$$P\left(y_i \mid v_i = 1, r_i = -2\right) P\left(v_i = 1, r_i = -2\right) = \frac{0,25}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i+2)^2}{2\sigma^2}}, \tag{5.12}$$

$$P\left(y_i \mid v_i = 1, r_i = +2\right) P\left(v_i = 1, r_i = +2\right) = \frac{0,25}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i-2)^2}{2\sigma^2}}, \tag{5.13}$$

$$P\left(y_i \mid v_i = 0\right) P\left(v_i = 0\right) = \frac{0,5}{\sigma\sqrt{2\pi}} e^{\frac{-y_i^2}{2\sigma^2}}. \tag{5.14}$$

Para podermos utilizar o algoritmo da mesma forma como ele fora utilizado anteriormente, precisamos obter $P_i = P\left(v_i = 1 \mid y_i\right)$. Para isso vamos utilizar novamente a regra de Bayes, a partir da qual temos:

$$1 - P_i = P\left(v_i = 0 \mid y_i\right) = \frac{P\left(y_i \mid v_i = 0\right) P\left(v_i = 0\right)}{P\left(y_i\right)}. \tag{5.15}$$

Pelo teorema das probabilidades totais, utilizando as Equações 5.12, 5.13 e 5.14 temos:

$$1\text{-}P_i = \frac{\frac{0,5}{\sigma\sqrt{2\pi}} e^{\frac{-y_i^2}{2\sigma^2}}}{\frac{0,5}{\sigma\sqrt{2\pi}} e^{\frac{-y_i^2}{2\sigma^2}} + \frac{0,25}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i+2)^2}{2\sigma^2}} + \frac{0,25}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i-2)^2}{2\sigma^2}}} \quad . \tag{5.16}$$

Fazendo as devidas simplificações temos:

$$1\text{-}P_i = \frac{1}{1+0,5e^{\frac{-(y_i+2)^2-y_i^2}{2\sigma^2}} + 0,5e^{\frac{-(y_i-2)^2-y_i^2}{2\sigma^2}}},$$

$$1\text{-}P_i = \frac{1}{1+0,5e^{\frac{-4y_i-4}{2\sigma^2}} + 0,5e^{\frac{+4y_i-4}{2\sigma^2}}} = \frac{1}{1+0,5e^{\frac{-2}{\sigma^2}}\left(e^{\frac{2y_i}{\sigma^2}} + e^{-\frac{2y_i}{\sigma^2}}\right)},$$

Finalmente obtemos que:

$$P_i = P(v_i = 1 \mid y_i) = 1 - \frac{1}{1+e^{\frac{-2}{\sigma^2}}\cosh\left(\frac{2y_i}{\sigma^2}\right)}. \tag{5.17}$$

5.5 RESULTADOS DE SIMULAÇÃO

Em se tratando de gravação magnética, na literatura o comprimento da palavra-código tem sido padronizado em 4096 bits, ou seja, 4kbits. A taxa do código de $0,89$ também tem sido praticada [27]. Assim, projetamos um código LDPC $(4096,3648)$, que possui taxa igual a $0,89$.

Utilizando o resultado da Equação 5.17 no algoritmo SPA, foram realizadas simulações com o canal $1-D$. Na Figura 5.9 é apresentada a probabilidade de erro de bit (BER) versus SNR (em dB) para o código projetado, obtida a partir de simulação de Monte-Carlo. A curva de desempenho para o canal $1-D$ não codificado é também apresentada para servir como referência para comparação. Verifica-se que o desempenho do código é muito superior ao do caso não codificado, como previamente esperado. Mas analisando as curvas pode-se avaliar o desempenho do código como muito bom. Esta simulação foi realizada utilizando uma matriz de verificação de paridade criada pelo procedimento apresentado nas Seções 5.3 e 5.4. A permutação realizada não foi otimizada. Portanto, é possível que exista um código com o desempenho melhor do que o apresentado. Mas a busca da matriz **H** ótima

é bastante complexa, porque estamos trabalhando com matrizes na ordem de 4096×448. E somente podemos verificar o seu desempenho depois de uma longa simulação.

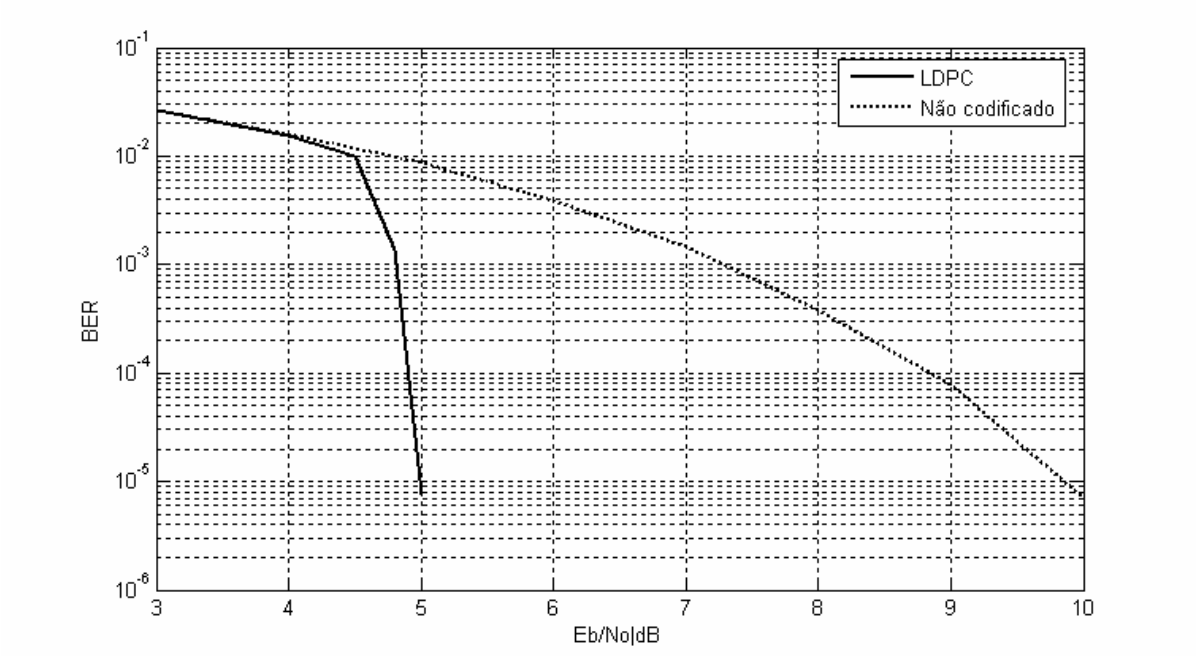


Figura 5.9: Gráfico comparativo entre o LDPC como proposto e o canal $1 - D$ sem codificação.

Observem na Figura 5.9 que a partir de 4.5dB o algoritmo passa a atuar. Para relação sinal-ruído muito baixa, o algoritmo não converge, que é uma característica comum a todos os códigos LDPC. Quando o algoritmo está atuando, obtemos resultados surpreendentes. Por exemplo, para 5dB de relação sinal-ruído, a probabilidade de erro de bit (BER) é inferior a 10^{-5}.

CAPÍTULO 6

CONCLUSÃO

Neste capítulo apresentaremos as conclusões tiradas no decurso deste trabalho, fazendo uma análise crítica dos resultados obtidos, e algumas propostas de trabalhos futuros.

O trabalho apresentado começou como um simples estudo de códigos LDPC, a partir do qual foi constatado que atualmente existem muitos trabalhos sobre esses códigos. Isso ocorre devido ao seu excelente desempenho, o que despertou o interesse na sua aplicação em novos sistemas tais como: a TV Digital e o WiMAX. A grande maioria desses trabalhos não aborda a codificação dos códigos LDPC. As soluções que encontramos para o codificador, em sua grande maioria, possuíam maior complexidade computacional do que o decodificador. Buscando resolver este problema, encontramos publicações sobre codificadores RA, uma sub-classe de códigos LDPC que apresenta complexidade linear tanto da codificação quando da decodificação. O fato de o processo de codificação RA envolver intrinsecamente uma certa pré-codificação fez nascer o interesse na aplicação desses códigos em canais de resposta parcial, para os quais a utilização de pré-codificação é uma prática comum. A partir daí começamos a elaborar o codificador RA, modificando-o para transformá-lo em um codificador para o canal $1-D$. Podemos perceber que o desempenho obtido foi suficientemente bom para se cogitar sua utilização em sistemas práticos. Esse desempenho ainda pode ser melhorado, através de utilização de outras taxas e pequenas modificações na técnica.

Os maiores problemas encontrados no decorrer deste trabalho foram na área de programação, devido à grande complexidade computacional exigida nas simulações. Todas as funções do codificador e do decodificador foram realizadas utilizando programação em linguagem C, e muitas passaram por várias etapas de otimização em função do enorme uso de processamento e memória.

6.1 TRABALHO FUTURO

Para trabalhos futuros, propomos as seguintes idéias:

- A utilização de outros algoritmos de decodificação para códigos LDPC, como por exemplo o Algoritmo SPA no domínio logarítmico (LSPA), e analisar o seu desempenho no canal $1-D$;

- A adoção de outras taxas do código LDPC, sobretudo taxas mais altas, observando a SNR a partir da qual o algoritmo passa a convergir e apresentar um bom desempenho;
- Considerar outros canais de resposta parcial. Para a gravação, por exemplo, canais descritos pelo polinômio $F(D)=(1-D)(1+D)^n$, onde $n=1,2$ e 3, podem ser considerados. A extensão da técnica proposta para esse caso mais geral não é difícil e é de grande interesse prático.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] C. E. Shannon, "A Mathematical Theory of Communication," Bell Systems Technical Journal, vol. 27, pp. 379-423 and 623-656, 1948.

[2] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[3] C. B. Schlegel and L. C. Pérez, *Trellis and Turbo Coding*, IEEE Press, Wiley-Interscience, 2004.

[4] L.Ping e W. K. Leung, "Decoding Low Density Parity Check Codes with Finite Quantization Bits", *IEEE Communications Letters*, vol. 4, nº 2, pp. 62-64, February 2000.

[5] R. G. Gallager, "*Low-Density Parity-Check Codes"*, Cambridge, MA: MIT Press, 1963.

[6] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes", *Electron. Lett.*, vol. 32, pp. 1645–1646, August 1996.

[7] T. J. Richardson and R. Urbanke, "The capacity of low-density paritycheck codes under message-passing decoding," *IEEE Trans. Inform.Theory*, vol. 47, February 2001.

[8] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Transactions Information Theory*, vol. IT-8, pp. 21-28, January 1962.

[9] D. J. C. MacKay, "Good Error-Correcting Codes based on Very Sparse Matrices", *IEEE Transactions on information Theory*, vol. 45, pp. 399-431, March 1999.

[10] D. J. C. MacKay and R. M. Neal, "Good Codes based on Very Sparse Matrices", *Cryptography and Coding. 5th IMA Conference ed. Colin Boyd, Lecture Notes in Computer Science* nº 1025, pp. 100-111, Springer, Berlim, 1995.

[11] R. M. Tanner, "A recursive approach to low complexity codes", *IEEE Transactions on Information Theory,* vol. IT-27, nº 5, pp. 533-547, September 1981.

[12] D. Divsalar, H. Jin, and R. J. McEliece. "Coding theorems for 'turbo-like' codes." *Proc. 36th Allerton Conf. on Communication, Control and Computing, Allerton, Illinois*, September 1998, pp. 201–210.

[13] G. Lechner e J. Sayir, "On the Convergence of Log-Likelihood Values in Iterative Decoding", *Mini-Workshop on Topics in Information Theory*, Essen, Alemanha, September 2002.

[14] Y. Mao e A. H. Banihashemi, "Decoding Low-Density Parity-Check Codes With Probabilistic Scheduling", *IEEE Communication Letters*, vol. 5, nº 10, pp. 414-416, October 2001.

[15] F. R. Kschischang e B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", *IEEE Journal on Selected Areas in Communications*, vol. 16, nº 2, pp. 219-230, February 1998.

[16] G. D. Forney, Jr., "On Iterative Decoding and the Two-Way Algorithm", *Proceedings of International Symposium on Turbo Codes and Related Topics*, Brest, França. pp. 12-25, September 1997.

[17] F. R. Kschischang, B. J. Frey e H. Loeliger, "Factor Graphs and the Sum-Product Algorithm", *IEEE Transactions on Information Theory*, vol. 47, nº 2, pp. 498-519, February 2001.

[18] F. R. Kschischang, "Codes Defined on Graphs", *IEEE Communications Magazine*, pp. 118-125, August 2003.

[19] T. Etzion, A. Trachtenberg e A. Vardy, "Which Codes Have Cycle-Free Tanner Graphs", *IEEE Transactions on Information Theory*, vol. 45, nº 6, pp. 2173-2181, September 1999.

[20] Y. Mao e A. H. Banihashemi, "A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lenghts", *IEEE ICC 2001*, Filândia, June 2001.

[21] J. A. McGowan e R. C. Williamson, "Loop Removal from LDPC Codes", *IEEE Information Theory Workshop 2003*, Paris, 2003.

[22] S. Chung, G. Forney, T. Richardson e R. Urbanke, "On the Design of Low Density Parity-Check Codes within 0.0045 dB of the Shannon Limit", *IEEE Communications Letters*, vol. 5, nº 2, pp. 58-60, February 2001.

[23] E. Lee and D. Messerschmitt, Digital Communication, 2nd ed. Norwell, MA: Kluwer, 1994.

[24] P. Kabal and S. Pasupathy, "Partial-response signaling," *IEEE Transactions Communications*, vol. COM-23, pp. 921–934, September 1975.

[25] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, Second Edition, Klumer Academic Publishers, Boston, 1988.

[26] B. F. Uchôa Filho, *Coding for some high density magnetic recording channel models*, Ph. D. dissertation, University of Notre Dame, Notre Dame, U.S.A., 1996.

[27] H. Song, R. M. Todd e J. R. Cruz, "Low Density Parity Check Codes for Magnetic Recording Channels", *IEEE Transactions on Magnetics*, vol. 36, no. 5, September 2000.